Association Rules & Frequent Itemsets

All you ever wanted to know about diapers, beers and their correlation!

The Market-Basket Problem

Given a database of transactions
where each transaction is a collection 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
Example: 30% of all transactions that contain diapers also contain beers; 5% of all transactions contain these items
- 30%: confidence of the rule
- 5%: support of the rule
We are interested in finding all rules, rather than verifying that a particular rule holds

What is Association Rule Mining?

- Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- Rule form: "Body ⊃ Head [support, confidence]"

Examples:
- \(\text{buy}(x, \text{"diapers") } \Rightarrow \text{buy}(x, \text{"beers") } [0.5\%, 60\%]\)
- \(\text{program}(x, \text{"TF") } \& \text{takes}(x, \text{"DM") } \Rightarrow \text{grade}(x, \text{"5") } [1\%, 75\%]\)

Market-Basket: Applications

- \(\ast \Rightarrow \text{Maintenance Agreement}\)
  - What should a store do to boost Maintenance Agreement sales?
- \(\text{Home Electronics } \Rightarrow \ast\)
  - What other products should a store stock up?
- Attached mailing in direct marketing
- Detecting "ping-pong"-ing of patients, faulty "collisions"

Rule Measures: Support and Confidence

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

Transaction ID| Items Bought
---|---
2000 | A, B, C
1000 | A, C
4000 | A, D
5000 | B, E, F

Let minimum support 50%, and minimum confidence 50%.
We have:
- \(A \Rightarrow C\) (50%, 66.6%)
- \(C \Rightarrow A\) (90%, 100%)
**Mining Association Rules—An Example**

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items Bought</th>
<th>Min. support 50%</th>
<th>Min. confidence 50%</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>

Frequent Itemset Support
- {A}: 75%
- {B}: 50%
- {C}: 50%
- {A, C}: 50%

For rule A → C:
- Support: \( \text{support}(\{A \Rightarrow C\}) = 50\% \)
- Confidence: \( \frac{\text{support}(\{A \Rightarrow C\})}{\text{support}(\{A\})} = 66.6\% \)

The Apriori principle: (large itemset property)

Any subset of a frequent itemset must also be frequent.

---

**Applications of Market-Basket Analysis**

- Supermarkets
  - Placement
  - Advertising
  - Sales
  - Coupons
- Many applications outside market basket data analysis
  - Prediction (telecom switch failure)
  - Web usage mining
- Many different types of association rules
  - Temporal
  - Spatial
  - Causal

---

**Aspect of Market Basket Mining**

- What is interesting?
- How do you make it run fast?
- Performance measured in
  - Number of database scans
  - Number of itemsets that must be counted

---

**What is Interesting? (first try)**

Set of items: \( I = \{ I_1, I_2, ..., I_m \} \)
Transactions: \( D = \{ t_1, t_2, ..., t_n \} \), \( t_j \subseteq I \)

Itemset \( L \) : set of items \( \{ I_{i_1}, I_{i_2}, ..., I_{i_k} \} \subseteq I \)

Support(L) : fraction of baskets that contain \( L \)

Large (Frequent) itemset: Itemset whose number of occurrences is above a threshold.

---

**Association Rules Example**

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>Bread, Jelly, Peanut Butter</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>Bread, Peanut Butter</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>Bread, Milk, Peanut Butter</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>Beer, Bread</td>
</tr>
<tr>
<td>( t_5 )</td>
<td>Beer, Milk</td>
</tr>
</tbody>
</table>

\( I = \{ \text{Beer, Bread, Jelly, Milk, Peanut Butter} \} \)

Support of {Bread, Peanut Butter} is 60%.

---

**Association Rule Definitions**

- **Association Rule (AR)**: implication \( X \Rightarrow Y \)
  where \( X, Y \subseteq I \) and \( X \cap Y = \emptyset \)
- **Support (s) of AR X \Rightarrow Y**: Percentage of transactions that contain \( X \cup Y \)
- **Confidence (c) of AR X \Rightarrow Y**: Ratio of number of transactions that contain \( X \cup Y \) to the number that contain \( X \)
Association Rules Example (cont’d)

<table>
<thead>
<tr>
<th>X ⇒ Y</th>
<th>s</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread ⇒ PeanutButter</td>
<td>60%</td>
<td>75%</td>
</tr>
<tr>
<td>PeanutButter ⇒ Bread</td>
<td>60%</td>
<td>100%</td>
</tr>
<tr>
<td>Beer ⇒ Bread</td>
<td>20%</td>
<td>50%</td>
</tr>
<tr>
<td>PeanutButter ⇒ Jelly</td>
<td>20%</td>
<td>33.3%</td>
</tr>
<tr>
<td>Jelly ⇒ PeanutButter</td>
<td>20%</td>
<td>100%</td>
</tr>
<tr>
<td>Jelly ⇒ Milk</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Association Rule Problem

Given a set of items I = \{I₁, I₂, ..., Iₙ\} and a database of transactions D = \{t₁, t₂, ..., tₙ\} where \(t_i = \{I_{x₁}, I_{x₂}, ..., I_{xₙ}\}\) and \(I_k \in I\), the Association Rule Problem is to identify all association rules \(X \Rightarrow Y\) where \(X, Y \subseteq I\) with a minimum support and confidence.

**NOTE:** Support of \(X \Rightarrow Y\) is same as support of \(X \cup Y\).

Finding Association Rules: A two step process

1. Find large (frequent) itemsets.
2. Generate rules from frequent itemsets.

Large Itemset: A Downwards-Closed Property

Apriori Example

<table>
<thead>
<tr>
<th>Pass</th>
<th>Candidates</th>
<th>Large Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Beer}, {Bread}, {Jelly}, {Milk}, {PeanutButter}</td>
<td>{Beer}, {Bread}, {Milk}, {PeanutButter}</td>
</tr>
<tr>
<td>2</td>
<td>{Beer, Bread}, {Beer, Milk}, {Beer, PeanutButter}, {Bread, Milk}, {Bread, PeanutButter}, {Milk, PeanutButter}</td>
<td>{Bread, PeanutButter}</td>
</tr>
</tbody>
</table>

How to find Itemsets with High Support?

- Find all itemsets with support > s

1-itemset: itemset with 1 item ...

k-itemset: itemset with k items

large itemset: itemset with support > s
candidate itemset: itemset that may have support > s
**Apriori Algorithm**

- start with all 1-itemsets
- go through data and count their support and find all "large" 1-itemsets
- combine them to form "candidate" 2-itemsets
- go through data and count their support and find all "large" 2-itemsets
- combine them to form "candidate" 3-itemsets...

**General Strategy**

**Step I:** Find all itemsets with \textit{minimum support (mainsup)}

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,3)</td>
<td>0.25</td>
</tr>
<tr>
<td>(1,2,3)</td>
<td>0.5</td>
</tr>
<tr>
<td>(2,3,5)</td>
<td>0.75</td>
</tr>
</tbody>
</table>

**Step II:** Generate rules from \textit{mainsup}ated itemsets

<table>
<thead>
<tr>
<th>Support</th>
<th>Confidence</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>66%</td>
<td>(1)</td>
</tr>
<tr>
<td>0.75</td>
<td>100%</td>
<td>(2)</td>
</tr>
</tbody>
</table>

**Algorithm to Guess Itemsets**

- Naïve way:
  - Extend all itemsets with all possible items
- More sophisticated:
  - Join \( L_k \) with itself, adding only a single, final item
    e.g.: \( (1 \ 2 \ 3), (1 \ 2 \ 4), (1 \ 3 \ 4), (2 \ 3 \ 4) \) produces \( (1 \ 2 \ 3 \ 4) \) and \( (1 \ 3 \ 4 \ 5) \)
  - Remove itemsets with an unsupported subset
    e.g.: \( (1 \ 3 \ 4 \ 5) \) has an unsupported subset \( (1 \ 4 \ 5) \)
    if \( \text{mainsup} = 50\% \)
    - Use the database to further refine \( C_k \)

**Run Time**

- \( k \) passes over data where \( k \) is the size of the largest candidate itemset
- Memory chunking algorithm \( \Rightarrow 2 \) passes over data on disk but multiple in memory

Toivonen 1996 gives a statistical technique which requires \( 1 + e \) passes (but more memory)

Brin 1997 - Dynamic Itemset Counting \( \Rightarrow 1 + e \) passes (less memory)

**Step I: Finding MinSup Itemsets**

- **Key fact:** Adding items to an itemset never increases its support
- **General Strategy:** Proceed inductively on itemset size
- **Apriori Algorithm:**
  1. Base case: Begin with all minsup itemsets of size 1 \( L_1 \)
  2. Without peeking at the DB, generate candidate itemsets of size \( k \) \( (C_k) \) from \( L_{k-1} \)
  3. Remove candidate itemsets that contain unsupported subsets
  4. Further refine \( C_k \) using the database to produce \( L_k \)

**Mining Frequent Itemsets: the Key Step**

- Find the \textit{frequent itemsets} the sets of items that have minimum support
  - Any 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 \( k \) \( (k\text{-itemset}) \)
- Use the frequent itemsets to generate association rules.
### The Apriori Algorithm

- **Join Step:** \( C_k \) is generated by joining \( L_k \) with itself.
- **Prune Step:** Any \((k-1)\)-itemset that is not frequent cannot be a subset of a frequent \( k \)-itemset.

**Pseudo-code:**

\( C_k \) Candidate itemset of size \( k \\
L_k \) frequent itemset of size \( k \\
for (k+1 \leq l \leq \log_2 \text{minsup}) \) do begin

\( C_{k+1} \) 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 \( t \)

\( L_{k+1} \) candidates in \( C_k \) with min support
end

return \( \cup_b L_b \)

### Apriori: Formulation from Original Paper

1. \( L_1 = \{ \text{large 1-itemsets} \} \)
2. for \( k = 2; L_{k-1} \neq \emptyset; k++ \) do begin
3. \( C_k = \text{aprior-gen}(L_{k-1}, \text{\textit{sup}}) \) // New candidates
4. forall transactions \( t \in D \) do begin
5. \( C_t = \text{subset}(C_k, t) \) // Candidates contained in \( t \)
6. forall candidates \( c \in C_t \) do
7. \( c\cdot\text{count}++ \)
8. end
9. \( L_k = \{ c \in C_k | c\cdot\text{count} \geq \text{minsup} \} \)
10. end
11. Answer = \( \bigcup_k L_k \)

### A-Priori Algorithm Example (\( s = 50\% \))

**Database D**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

**\( L_1 \)**

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

**Scan D**

**\( L_2 \)**

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2}</td>
<td>1</td>
</tr>
<tr>
<td>{1,3}</td>
<td>2</td>
</tr>
<tr>
<td>{1,5}</td>
<td>1</td>
</tr>
<tr>
<td>{2,3}</td>
<td>2</td>
</tr>
<tr>
<td>{2,5}</td>
<td>3</td>
</tr>
<tr>
<td>{3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>

**\( C_2 \)**

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2}</td>
<td>1</td>
</tr>
<tr>
<td>{1,3}</td>
<td>2</td>
</tr>
<tr>
<td>{1,5}</td>
<td>1</td>
</tr>
<tr>
<td>{2,3}</td>
<td>2</td>
</tr>
<tr>
<td>{2,5}</td>
<td>3</td>
</tr>
<tr>
<td>{3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>

**Scan D**

**\( L_3 \)**

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2,3}</td>
<td>1</td>
</tr>
</tbody>
</table>

**\( C_3 \)**

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2,3,5}</td>
<td>2</td>
</tr>
</tbody>
</table>

### Apriori: How to Generate Candidates?

**STEP 1: Self-join operation**

- insert into \( C_k \)
- select \( p, \text{item}_1, \text{item}_2, \ldots, \text{item}_{k-1}, q, \text{item}_{k-1} \)
- from \( L_{k-1} \) \& \( L_{k-1} \)
- where \( p, \text{item}_1 = q, \text{item}_2 = q, \text{item}_{k-1} = q, \text{item}_{k-1} < q, \text{item}_{k-1} \)

**STEP 2: Subset filtering**

- forall itemsets \( c \in C_k \) do
- forall \((k-1)\)-subsets \( s \) of \( c \) do
- if \( s \notin L_{k-1} \) then
- delete \( c \) from \( C_k \)

### Example of Generating Candidate Itemsets

- \( L_2 = \{ abc, abd, acd, ace, bcd\} \)
- Self-joining: \( L_2 \cdot L_2 \)
  - \( abcd \) from \( abc \) and \( abd \)
  - \( acde \) from \( acd \) and \( ace \)

- Pruning based on the \( \alpha \)-priori principle:
  - \( acde \) is removed because \( acde \) is not in \( L_2 \)
- \( C_2 = \{ abcd \} \)
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-itemsets 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.

Is Apriori Fast Enough? — Performance Bottlenecks

- The core of the Apriori algorithm:
  - Use frequent (k-1) itemsets to generate candidate frequent k-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of Apriori candidate generation
  - Huge candidate sets:
    - 10^4 frequent 1-itemsets will generate 10^7 candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., (a_1, a_2, ..., a_100), one needs to generate 2^100 = 10^30 candidate sets.
  - Multiple scans of database:
    - Needs (n+1) scans, where n is the length of the longest pattern.

Step II: Generating Rules

Key fact:

Moving items from the antecedent to the consequent never changes support and never increases confidence.

Algorithm

- For each itemset IS with minsup:
  - Find all minconf rules with a single consequent of the form (IS - L_J ⇒ L_i)
  - Guess candidate consequents Cl by appending items from IS - L_{k-1} to L_{k-2}
  - Verify confidence of each rule IS - Cl ⇒ C_j using known itemset support values

Algorithm to Generate Association Rules

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

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

ARGen Algorithm:

1. \( R \leftarrow \emptyset \)
2. for each \( l \in L \) do
   for each \( x \in I \) such that \( x \neq \emptyset \) and \( x \neq l \) do
      if \( \text{support}(x) \geq s \) then
         \( R \leftarrow R \cup \{ x \colon (l - x) \} \)

Questions

- How are rules ranked?
- Do the minsup and minconf find interesting rules?
- Do they omit any interesting rules?
- What about maximum support?
- How well will this approach work for other problems (e.g., clustering, classification)?
But what is really interesting?

- \( A \implies B \)
- \( \text{Support} = P(AB) \)
- \( \text{Confidence} = P(B|A) \)
- \( \text{Interest} = P(AB)/P(A)P(B) \)
- \( \text{Implication Strength} = P(A)P(\neg B)/P(\neg A\neg B) \)

Interestingness Measurements

**Objective measures**
- Two popular measurements:
  - \( \text{support} \) and \( \text{confidence} \)

**Subjective measures** (Silberschatz & Tuzhilin, KDD95)
- A rule (pattern) is interesting if
  - it is unexpected( surprising to the user); and/or
  - actionable (the user can do something with it)

Criticism to Support and Confidence

Example 1: (Aggarwal & Yu, paper at PODS98)

- Among 10000 students
  - 6000 play basketball
  - 7500 eat cereal
  - 4000 both play basketball and eat cereal
- play basketball \( \implies \) eat cereal (40%, 66.7%) is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
- play basketball \( \implies \) not eat cereal (20%, 33.3%) is far more accurate, although with lower support and confidence

<table>
<thead>
<tr>
<th></th>
<th>basketball</th>
<th>not basketball</th>
<th>sum(col.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cereal</td>
<td>4000</td>
<td>3500</td>
<td>7500</td>
</tr>
<tr>
<td>not cereal</td>
<td>2000</td>
<td>500</td>
<td>2500</td>
</tr>
<tr>
<td>sum(rows)</td>
<td>6000</td>
<td>4000</td>
<td>10000</td>
</tr>
</tbody>
</table>

Criticism to Support and Confidence (Cont.)

Example 2:

- \( X \) and \( Y \) positively correlated,
- \( X \) and \( Z \) negatively related
- support and confidence of \( X \implies Z \) dominates

We need a measure of dependent or correlated events

\[
corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}
\]

- \( P(B|A)/P(B) \) is also called the lift of rule \( A \implies B \)

Other Interestingness Measures: Interest

- Interest (correlation, lift) \( \frac{P(A \cup B)}{P(A)P(B)} \)
  - taking both \( P(A) \) and \( P(B) \) in consideration
  - \( P(A \cup B) = P(B)^{*}P(A) \), if \( A \) and \( B \) are independent events
  - \( A \) and \( B \) negatively correlated, if the value is less than 1;
  - otherwise \( A \) and \( B \) positively correlated

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>60%</td>
<td>2</td>
</tr>
<tr>
<td>( Y )</td>
<td>37.5%</td>
<td>0.9</td>
</tr>
<tr>
<td>( Z )</td>
<td>12.5%</td>
<td>0.57</td>
</tr>
</tbody>
</table>

But what is really really interesting?

- Causality
- Surprise