Lec 05–06: Association Rules and Pattern Mining Algorithms

COSC–254 – February 11–13, 2019
Outline

Recap about Frequent Itemsets

Association Rules
- Definitions
- Measures of interestingness and their properties
- Mining task
- Algorithm

Algorithms to mine the Frequent Itemsets
- Apriori
- Eclat
Recap: Definitions

- **Items** $\mathcal{I}$
- **Dataset** $\mathcal{D} = \{t_1, \ldots, t_n\}$ of transactions, $t_i \subseteq \mathcal{I}$, $i = 1, \ldots, n$.
- **Itemset** $A \subseteq \mathcal{I}$
  - support of $A$ in $\mathcal{D}$: $\text{supp}_\mathcal{D}(A) = |\{t \in \mathcal{D} : A \subseteq t\}|$.
  - frequency of $A$ in $\mathcal{D}$: $\text{freq}_\mathcal{D}(A) = \frac{\text{supp}_\mathcal{D}(A)}{n}$
- Given a minimum support threshold $\ell$, the Frequent Itemsets in $\mathcal{D}$ w.r.t. $\ell$:
  $$\text{FI}(\mathcal{D}, \ell) = \{A \subseteq \mathcal{I} : \text{supp}_\mathcal{D}(A) \geq \ell\}$$

For a minimum frequency threshold $\theta$,
$$\text{FI}(\mathcal{D}, \theta) = \{A \subseteq \mathcal{I} : \text{freq}_\mathcal{D}(A) \geq \theta\}$$
Recap: the Itemset Lattice and the Naïve Search Strategy

Naïve mining strategy:
Traverse the lattice (BFS or DFS), and compute support for each visited itemset.

Time to get $\text{supp}_D(A)$: $O(|I| |D|)$ ($O(|I|)$ to check whether $t \in D$ contains $A$)

Total mining time for the naïve strategy: $O(2^{|I|} |I| |D|)$

Figure from J. Vreeken.
Recap: Anti-Monotonicity Property of the Support

Theorem (Anti-monotonicity of the support)

For any $A \subset B \subseteq \mathcal{I}$, it holds

\[ \text{supp}_D(A) \geq \text{supp}_D(B) \]

A.k.a. downward closure

The anti-monotonicity gives us a possible way to prune the search space.

Corollary (Sufficient condition for pruning)

For any $B \subseteq \mathcal{I}$,

if $\exists A \subset B$ s.t. $A \notin \text{Fl}(D, \ell)$, then $B \ldots \notin \text{Fl}(D, \ell)$.
Recap: the Negative Border

Minimum support in the figure: $\ell = 2$

Finding the **Negative Border** is equivalent to finding the FIs.

Figure from J. Vreeken.
Outline

✓ Recap about Frequent Itemsets

Association Rules
• Definitions
• Measures of interestingness and their properties
• Mining task
• Algorithm

Algorithms to mine the Frequent Itemsets
• Apriori
• Eclat
Definition (Association Rule)

An expression in the form \( A \Rightarrow B \)

for \( A, B \subset I, A \cap B = \emptyset \)

**INTERPRETATION:** Transactions containing \( A \) also *often* contain \( B \).

**EXAMPLE:** customers buying a camera also *often* buy a memory card and a battery.
Search space of ARs

(AR: $A \Rightarrow B$, $A, B \subset \mathcal{I}$, $A \cap B = \emptyset$)

Q: Given $\mathcal{I}$, $|\mathcal{I}| = m$, how many possible ARs are there?

A: Possible ARs: $\sum_{k=2}^{m} \left[ \binom{m}{k} (2^k - 2) \right] \approx O(2^{2m})$

Must define an interestingness measure to assign a score to each AR:

Interestingness measure $g : ARs \rightarrow [0, 1]$

We can then use $g$ to find the most interesting ARs (w.r.t. $g$).

The measure $g$ we define will be a combination of two measures.
Interestingness measure for ARs

\[(A, B \subseteq I, A \cap B = \emptyset)\]

\[A \Rightarrow B = \text{“when } A \subseteq t \in \mathcal{D}, \text{ often also holds } B \subset t\]" 

\[A \subseteq t \land B \subseteq t \iff A \cup B \subseteq t\]

If \(\text{supp}_\mathcal{D}(A \cup B)\) is small, \(A \Rightarrow B\) is not interesting.

Definition (Support of AR \(A \Rightarrow B\) in \(\mathcal{D}\))

\[\text{supp}_\mathcal{D}(A \Rightarrow B) = \text{supp}_\mathcal{D}(A \cup B)\]

The support takes part in defining an interestingness measure for ARs.
Interestingness measure for ARs

\((A, B \subseteq \mathcal{I}, A \cap B = \emptyset)\)

\[ A \Rightarrow B = \text{“ when } A \subseteq t \in \mathcal{D}, \text{ often also holds } B \subset t \text{”} \]

For interesting ARs, it should hold

\[ \text{supp}_D(A \cup B) \approx \text{supp}_D(A) \]

The closest the supports are, the more interesting \(A \Rightarrow B\)

**Definition (Confidence of AR \(A \Rightarrow B\) in \(\mathcal{D}\))**

\[ \text{conf}_D(A \Rightarrow B) = \frac{\text{supp}_D(A \cup B)}{\text{supp}_D(A)} \in [0, 1] \]

Interesting rules should have *high confidence*. 
Example

\[ t_1 = \{m, c, b\}, \quad t_2 = \{m, p, j\}, \quad t_3 = \{m, b\}, \quad t_4 = \{c, j\}, \]
\[ t_5 = \{m, b, p\}, \quad t_6 = \{m, c, b, j\}, \quad t_7 = \{c, b, j\}, \quad t_8 = \{b, c\} \]

AR \{m, b\} \Rightarrow \{c\}

Support: \( \text{supp}_D(\{m, b\} \Rightarrow \{c\}) = \text{supp}_D(\{m, b, c\}) = 2 \)

Confidence: \( \text{conf}_D\{m, b\} \Rightarrow \{c\} = \frac{2}{4} = 0.5 \)

Example from slides at \text{http://mmds.org}. \[ 12 \]
The AR mining task

Given a dataset $\mathcal{D}$ and

- a minimum \textit{support} threshold $\ell$ (or min. freq. thres. $\theta$); and
- a minimum \textit{confidence} threshold $\gamma \in [0, 1]$

Find

$$\text{AR}(\mathcal{D}, \ell, \gamma) = \{ A \Rightarrow B : \text{supp}_D(A \Rightarrow B) \geq \ell \land \text{conf}_D(A \Rightarrow B) \geq \gamma \}$$

The search space is huge, we need a \textit{smart pruning strategy}. 
Pruning strategy for ARs

Only rules \( A \Rightarrow B \) s.t. \( \text{supp}_D(A \Rightarrow B) = \text{supp}_D(A \cup B) \geq \ell \) may be in \( \text{AR}(D, \ell, \gamma) \).

**Lemma (Sufficient condition for pruning ARs)**

If \( \text{supp}_D(A \cup B) < \ell \), then \( \text{supp}_D(A \Rightarrow B) < \ell \) and
\[
A \Rightarrow B \notin \text{AR}(D, \ell, \gamma)
\]

Knowing that \( \text{supp}_D(A \cup B) < \ell \) allow us to prune \( 2^{|A \cup B|} - 2 \) ARs.

Q: What are the \( G \subseteq \mathcal{I} \) s.t. \( \text{supp}_D(G) \geq \ell \)?

A: \( \text{FI}(D, \ell) \).

Idea for finding \( \text{AR}(D, \ell, \gamma) \):

First find \( \text{FI}(D, \ell) \), then use it to generate rules, and check their confidence.
Generating & pruning ARs

Q: How to generate and prune rules in a smart way?

A: Study properties of the confidence

Theorem

For any non-empty \( A, B \subseteq I \), such that \( A \cap B = \emptyset \), it holds

\[
\text{conf}_D(A \cup \{a\} \Rightarrow B \setminus \{a\}) \geq \text{conf}_D(A \Rightarrow B)
\]

for any \( a \in B \).

Proof: HW03

Corollary (Sufficient condition for pruning ARs)

If \( \exists a \in I, a \in B \ s.t. \ \text{conf}_D(A \cup \{a\} \Rightarrow B \setminus \{a\}) < \gamma \), then

\[
\text{conf}_D(A \Rightarrow B) < \gamma
\]
Corollary (Sufficient condition for pruning ARs)

If $\exists a \in \mathcal{I}, a \in B$ s.t. $\text{conf}_D(A \cup \{a\} \Rightarrow B \setminus \{a\}) < \gamma$, then $\text{conf}_D(A \Rightarrow B) < \gamma$

**Example:**

$A = \{\text{camera}\}$, $B = \{\text{battery, mem. card}\}$

$\text{supp}_D(A) = 6$, $\text{supp}_D(B) = 2$, $\text{supp}_D(A \cup \{a\}) = 4$, $\text{supp}_D(A \cup B) = 2$

$\text{conf}_D(A \cup \{a\} \Rightarrow B \setminus \{a\}) = \frac{2}{4} = 0.5$

$\text{conf}_D(A \Rightarrow B) = \frac{2}{6} = 0.3$
Generating & pruning ARs

Let’s look at the search space.

**Idea for smart search strategy:**

1. Generate rules with short r.h.s., and check their confidence;
2. Generate a rule with larger r.h.s. only if all related “shorter” rules have confidence $\geq \gamma$
Algorithm for mining ARs

**INPUT**: dataset $\mathcal{D}$, min. supp. thres. $\ell$, min. conf. thres. $\gamma$

**OUTPUT**: $\text{AR}(\mathcal{D}, \ell, \gamma)$

0. $Z \leftarrow \emptyset$ // to keep the interesting ARs

1. Mine $\text{FI}(\mathcal{D}, \ell)$ // coming soon

2. For each $G \in \text{FI}(\mathcal{D}, \ell)$ do
   
   2.0 For each $a \in G$, if $\text{conf}_\mathcal{D}(G \{a\} \Rightarrow \{a\}) \geq \gamma$, then $Z \leftarrow Z \cup \{G \{a\} \Rightarrow \{a\}\}$

   2.1 For each $A \subset G$ s.t. $1 \leq |A| < |G| - 2$ do // from long to short $A$
      
      2.1.a $B \leftarrow G \setminus A$

      2.1.b If, for each $a \in B$, “$A \cup \{a\} \Rightarrow B \setminus \{a\}$” is in $Z$, then

      2.1.b.i If $\text{conf}_\mathcal{D}(A \Rightarrow B) \geq \gamma$, then $Z \leftarrow Z \cup \{A \Rightarrow B\}$

3. Return $Z$
Example

Example from slides at http://mmds.org.

\[ t_1 = \{m, c, b\}, \quad t_2 = \{m, p, j\}, \quad t_3 = \{m, c, b, n\}, \quad t_4 = \{c, j\}, \]
\[ t_5 = \{m, b, p\}, \quad t_6 = \{m, c, b, j\}, \quad t_7 = \{c, b, j\}, \quad t_8 = \{b, c\} \]

Fix \( \ell = 3, \gamma = 0.75 \)

1) \( \text{FI}(\mathcal{D}, \ell) = \{\{b\}, \{m\}, \{c\}, \{j\}, \{b, m\}, \{b, c\}, \{c, m\}, \{c, j\}, \{m, c, b\}\} \)

2) Generate rules: E.g., for \( G = \{m, c, b\} \) (\( \text{supp}_\mathcal{D}(\{m, c, b\}) = 3 \)):

\( \{b, m\} \Rightarrow \{c\}: c = 3/4 \checkmark, \quad \{c, m\} \Rightarrow \{b\}: c = 3/3 \checkmark, \quad \{b, c\} \Rightarrow \{m\}: c = 3/5 \times, \)

We then know that \( \{b\} \Rightarrow \{m, c\} \) and \( \{c\} \Rightarrow \{m, b\} \) have conf. \( \leq 3/5 \times. \)

We need to check \( \{m\} \Rightarrow \{b, c\}: c = 3/5 \times. \)
AR Recap

- $A \Rightarrow B$: transactions containing $A$ also often contain $B$
- $\text{conf}_D(A \Rightarrow B)$: confidence of $A \Rightarrow B$ in $D$
- $\text{AR}(D, \ell, \gamma)$: interesting ARs
- Properties of the confidence to prune the search space
- Algorithm to mine ARs

Outline

✓ Recap about Frequent Itemsets

✓ Association Rules
  • Definitions
  • Measures of interestingness and their properties
  • Mining task
  • Algorithm

Algorithms to mine the Frequent Itemsets
  • Apriori
  • Eclat

Different search strategies
Intuition behind Apriori

Task: Given dataset $\mathcal{D}$ and min. supp. thres. $\ell$, find $\text{FI}(\mathcal{D}, \ell)$.

**Intuition for Apriori:**

- *Level-wise exploration* of the lattice (BFS):
  go from low levels (short itemsets) to high levels (long itemsets)
- Not all itemsets are *candidate* FIs:
  use *antimonotonicity* to avoid generating and checking the support of any itemset that has an infrequent subset.
Intuition behind Apriori

(“Count” here means “Check the support of”)

Figure from slides at http://mmds.org.
Apriori

INPUT: dataset $D$, min. supp. thres. $\ell$
OUTPUT: $\FI(D, \ell)$

$F_1 \leftarrow \{\{i\} : i \in \mathcal{I} \text{ s.t. } \supp_D(\{i\}) \geq \ell\}$ \hspace{1em} // Frequent items

$k \leftarrow 1$

While $F_k \neq \emptyset$ do

// Use $F_k$ to generate candidates of length $k + 1$

$C_{k+1} \leftarrow \{A \subseteq \mathcal{I} : |A| = k + 1 \land \forall E \subset A, |E| = k, E \in F_k\}$

$F_{k+1} \leftarrow \emptyset$

For each $A \in C_{k+1}$ do

If $\supp_D(A) \geq \ell$ then $F_{k+1} \leftarrow F_{k+1} \cup \{A\}$

$k \leftarrow k + 1$

Return $F_1 \cup F_2 \cup \cdots \cup F_{k-1}$
Apriori in action

data

itemset lattice

Images by J. Vreeken
Apriori in action

Images by J. Vreeken

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**data**

**itemset lattice**
Apriori in action

Data

Itemset lattice

Images by J. Vreeken
Apriori in action

Images by J. Vreeken

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

data

itemset lattice

Images by J. Vreeken
Apriori in action

Not generated as candidates, because subset cd is infrequent!

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

data

itemset lattice

Images by J. Vreeken
Improving the I/O of Apriori

From the pseudocode:

For each \( A \in C_{k+1} \) do
   If \( \text{supp}_D(A) \geq \ell \) then \( F_{k+1} \leftarrow F_{k+1} \cup \{A\} \)

Q: How do we compute the support of \( A \)?

For each \( A \in C_{k+1} \) do
   \( s_A \leftarrow 0 \)
   For each \( t \in D \), if \( A \subseteq t \), \( s_A \leftarrow s_A + 1 \)
   If \( s_A \geq \ell \) then \( F_{k+1} \leftarrow F_{k+1} \cup \{A\} \)

A scan of the dataset for each candidate is too expensive (up to \( O(2^{|I|}) \) scans!)
Improving the I/O of Apriori

For each $A \in C_{k+1}$ do

$S_A \leftarrow 0$

For each $t \in D$, if $A \subseteq t$, $s_A \leftarrow s_A + 1$

If $s_A \geq \ell$ then $F_{k+1} \leftarrow F_{k+1} \cup \{A\}$

How to speed it up? Switch the order of the loops!

Replace the lines above with the following:

$S_A \leftarrow 0$ for each $A \in C_{k+1}$

For each $t \in D$ do

For each $A \subseteq t$, $|A| = k + 1$ do

If $A \in C_{k+1}$ then $S_A \leftarrow S_A + 1$

$F_{k+1} \leftarrow \{A \in C_{k+1} : S_A \geq \ell\}$

Only one dataset scan per level, so $O(|D|)$ scans.
Outline

☑ Recap about Frequent Itemsets

☑ Association Rules
  • Definitions
  • Measures of interestingness and their properties
  • Mining task
  • Algorithm

Algorithms to mine the Frequent Itemsets
  • ✔ Apriori
  • Eclat

} Different search strategies
Eclat motivation

Q: Where does Apriori spend most of its time?
A: Computing the supports of itemsets

Eclat speed ups the computation of supports with *tidsets*

**Definition (Tidset of \( A \subseteq \mathcal{I} \))**

For any \( A \subseteq \mathcal{I} \), the *tidset* \( t(A) \) is the set of *transactions IDs* of the transactions containing \( A \):

\[
t(A) = \{ \text{tid} : (\text{tid}, t) \in \mathcal{D}, A \subseteq t \}
\]

Tidsets are *indices*, i.e., data structures that allow *fast access* to the rows (transactions) containing an itemset.
Tidsets

**Definition:** 
\[ t(A) = \{ \text{tid} : (\text{tid}, t) \in D, A \subseteq t \} \]

**Fact**

*It holds* 
\[ |t(A)| = \text{supp}_D(A). \]

**Lemma**

*For any* \( A, B \subseteq I \), *it holds* 
\[ t(A \cup B) = t(A) \cap t(B). \]
Example of tidsets

**Definition:** \( t(A) = \{ \text{tid} : (\text{tid}, t) \in \mathcal{D}, A \subseteq t \} \)

**Lemma:** For any \( A, B \subseteq \mathcal{I} \), it holds \( t(A \cup B) = t(A) \cap t(B) \).

**Dataset:**

\[
\begin{align*}
t_1 &= \{m, c, b\}, & t_2 &= \{m, p, j\}, & t_3 &= \{m, b\}, & t_4 &= \{c, j\}, \\
t_5 &= \{m, b, p\}, & t_6 &= \{m, c, b, j\}, & t_7 &= \{c, b, j\}, & t_8 &= \{b, c\} \\
\end{align*}
\]

\[
\begin{align*}
t(\{b, c\}) &= \{1, 6, 7, 8\} & t(\{m, c\}) &= \{1, 6\} \\
t(\{m, c, b\}) &= t(\{b, c\} \cup \{m, c\}) = t(\{b, c\}) \cap t(\{m, c\}) = \{1, 6\} \\
\end{align*}
\]
Eclat assumes that tidsets \( t(\{a\}) \) for each item \( a \in \mathcal{I} \) are available.

**Idea:** compute support of large itemsets by

1. *intersecting* tidsets of smaller itemsets; and
2. counting the cardinality of intersections.

Q: Of what subsets of \( A \) should we intersect the tidsets to obtain the support of \( A \)?
Prefix of an itemset

Additional assumption: $\mathcal{I}$ is *ordered*, itemsets are *ordered* accordingly:

$$\mathcal{I} = \{1, 2, 3, \ldots \}, \quad A = \{3, 7, 8\}, \quad B = \{9, 25\}, \quad \ldots$$

**Definition (Prefix of $A \subseteq \mathcal{I}$)**

Given $A \subseteq \mathcal{I}$, the *prefix* of $A$ is the subset of $A$ containing the first $|A| - 1$ items:

$$A = \{a_1, \ldots, a_{\ell-1}, a_\ell\}$$

E.g.: $A = \{\text{camera, battery, mem. card}\}$, prefix = $\{\text{camera, battery}\}$
Prefix equivalence class

Definition (Prefix equivalence class (PEC) for $A \subset \mathcal{I}$)

For any itemset $A \subset \mathcal{I}$, the prefix equivalence class $\text{PEC}(A)$ is the set of all itemsets with prefix $A$:

$$\text{PEC}(A) = \{ A \cup \{a\} : a \in \mathcal{I} \}$$

E.g.: $\mathcal{I} = \{1, \ldots, 5\}$, $A = \{2, 3\}$,

$$\text{PEC}(A) = \{ \{2, 3, 4\}, \{2, 3, 5\} \}$$

Q: Of what subsets of $A$ should we intersect the tidsets to obtain the support of $A$?

Eclat computes the support of $G = A \cup \{a\} \cup \{b\}$, $|G| = |A| + 2$ using the tidsets of $A \cup \{a\}$ and $A \cup \{b\}$, which are in $\text{PEC}(A)$:

$$\text{supp}_D(G) = |t(A \cup \{a\}) \cap t(A \cup \{b\})|$$

Eclat traverses the lattice in a depth-first search (DFS) way.
Eclat in action

First PEC with Ø as prefix

Images by J. Vreeken
Eclat in action

First PEC with $\emptyset$ as prefix

2nd PEC with $A$ as prefix

Images by J. Vreeken
First PEC with ∅ as prefix

2nd PEC with A as prefix

Infrequent!

Images by J. Vreeken
Eclat in action

First PEC with $\emptyset$ as prefix
Eclat in action

First PEC with ø as prefix

Images by J. Vreeken
Eclat in action

First PEC with \( \emptyset \) as prefix

Images by J. Vreeken
Eclat in action

First PEC with $\emptyset$ as prefix

This PEC only after everything starting with $A$ is done

Images by J. Vreeken
Eclat vs. Apriori

Apriori explores the lattice in a \textit{breadth-first search} (BFS) way.

Eclat in a \textit{depth-first search} (DFS) way.

Eclat uses \textit{smart data structures} (tidsets) to speed-up support computation. These structures are fast only if they all fit into main memory.

Apriori limits the number of full dataset scans to $O(|I|)$. It is often the fastest when the data does not fit in memory.