Chapter 9: Summarizing Itemsets
Maximal Frequent Itemsets

Given a binary database \( D \subseteq T \times I \), over the tids \( T \) and items \( I \), let \( F \) denote the set of all frequent itemsets, that is,

\[
F = \{ X \mid X \subseteq I \text{ and } \text{sup}(X) \geq \text{minsup} \}
\]

A frequent itemset \( X \in F \) is called \textit{maximal} if it has no frequent supersets. Let \( M \) be the set of all maximal frequent itemsets, given as

\[
M = \{ X \mid X \in F \text{ and } \not\exists Y \supset X, \text{ such that } Y \in F \}
\]

The set \( M \) is a condensed representation of the set of all frequent itemset \( F \), because we can determine whether any itemset \( X \) is frequent or not using \( M \). If there exists a maximal itemset \( Z \) such that \( X \subseteq Z \), then \( X \) must be frequent; otherwise \( X \) cannot be frequent.
An Example Database

Transaction database

Frequent itemsets ($\text{minsup} = 3$)

<table>
<thead>
<tr>
<th>Tid</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ABDE</td>
</tr>
<tr>
<td>2</td>
<td>BCE</td>
</tr>
<tr>
<td>3</td>
<td>ABDE</td>
</tr>
<tr>
<td>4</td>
<td>ABCE</td>
</tr>
<tr>
<td>5</td>
<td>ABCDE</td>
</tr>
<tr>
<td>6</td>
<td>BCD</td>
</tr>
</tbody>
</table>

$\text{sup}$ Itemsets

<table>
<thead>
<tr>
<th>sup</th>
<th>Itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>{B}</td>
</tr>
<tr>
<td>5</td>
<td>{E, BE}</td>
</tr>
<tr>
<td>4</td>
<td>{A, C, D, AB, AE, BC, BD, ABE}</td>
</tr>
<tr>
<td>3</td>
<td>{AD, CE, DE, ABD, ADE, BCE, BCE, ABDE}</td>
</tr>
</tbody>
</table>

$\text{C} = \{B, CE, BC, BD, AB, BCE\}$

$\text{M} = \{\frac{ABDE}{3}, BCE\}$

$\text{sup}(AB) \geq 3$
Given $T \subseteq \mathcal{T}$, and $X \subseteq \mathcal{I}$, define

$$t(X) = \{ t \in \mathcal{T} \mid t \text{ contains } X \}$$
$$i(T) = \{ x \in \mathcal{I} \mid \forall t \in T, \ t \text{ contains } x \}$$
$$c(X) = i \circ t(X) = i(t(X))$$

The function $c$ is a **closure operator** and an itemset $X$ is called **closed** if $c(X) = X$. It follows that $t(c(X)) = t(X)$. The set of all closed frequent itemsets is thus defined as

$$C = \{ X \mid X \in \mathcal{F} \text{ and } \not\exists Y \supset X \text{ such that } \text{sup}(X) = \text{sup}(Y) \}$$

$X$ is closed if all supersets of $X$ have strictly less support, that is, $\text{sup}(X) > \text{sup}(Y)$, for all $Y \supset X$.

The set of all closed frequent itemsets $C$ is a condensed representation, as we can determine whether an itemset $X$ is frequent, as well as the exact support of $X$ using $C$ alone.
Minimal Generators

A frequent itemset $X$ is a *minimal generator* if it has no subsets with the same support:

$$
G = \{ X \mid X \in \mathcal{F} \text{ and } \forall Y \subset X, \text{ such that } \sup(X) = \sup(Y) \}
$$

In other words, all subsets of $X$ have strictly higher support, that is, $\sup(X) < \sup(Y)$, for all $Y \subset X$.

Given an equivalence class of itemsets that have the same tidset, a closed itemset is the unique maximum element of the class, whereas the minimal generators are the minimal elements of the class.
Frequent Itemsets: Closed, Minimal Generators and Maximal

Itemsets boxed and shaded are closed, double boxed are maximal, and those boxed are minimal generators.
X = A
\[ t(A) = 1345 \]
\[ i(t(A)) = i(1345) = A \subseteq \]

\[ i(t(1345)) = i(1345) = A \subseteq \]

**Closure Operation:**
\[ i(t(x)) = c(x) \]

Closed Itemset:
\[ X = c(x) \]

\[ X \subseteq T \]

\[ X = A \]

A is not closed, but A\(\bar{B}C\) is closed.
Formal Concepts

Concept = closed itemset

\[(\text{intent} \iff \text{extent})\]

set of attributes

Set of objects

B - 123456
BD - 1356
BC - 2456
CT - 12345
AB - 1345
BC - 245
ABDE - 135

(ABC, 1345-)

Concept
Mining maximal itemsets requires additional steps beyond simply determining the frequent itemsets. Assuming that the set of maximal frequent itemsets is initially empty, that is, \( \mathcal{M} = \emptyset \), each time we generate a new frequent itemset \( X \), we have to perform the following maximality checks:

- **Subset Check:** \( \exists Y \in \mathcal{M}, \text{ such that } X \subset Y \). If such a \( Y \) exists, then clearly \( X \) is not maximal. Otherwise, we add \( X \) to \( \mathcal{M} \), as a potentially maximal itemset.

- **Superset Check:** \( \exists Y \in \mathcal{M}, \text{ such that } Y \subset X \). If such a \( Y \) exists, then \( Y \) cannot be maximal, and we have to remove it from \( \mathcal{M} \).
GenMax Algorithm: Maximal Itemsets

GenMax is based on dEclat, i.e., it uses diffset intersections for support computation. The initial call takes as input the set of frequent items along with their tidsets, \( \langle i, t(i) \rangle \), and the initially empty set of maximal itemsets, \( \mathcal{M} \). Given a set of itemset–tidset pairs, called IT-pairs, of the form \( \langle X, t(X) \rangle \), the recursive GenMax method works as follows.

If the union of all the itemsets, \( Y = \bigcup X_i \), is already subsumed by (or contained in) some maximal pattern \( Z \in \mathcal{M} \), then no maximal itemset can be generated from the current branch, and it is pruned. Otherwise, we intersect each IT-pair \( \langle X_i, t(X_i) \rangle \) with all the other IT-pairs \( \langle X_j, t(X_j) \rangle \), with \( j > i \), to generate new candidates \( X_{ij} \), which are added to the IT-pair set \( P_i \).

If \( P_i \) is not empty, a recursive call to GenMax is made to find other potentially frequent extensions of \( X_i \). On the other hand, if \( P_i \) is empty, it means that \( X_i \) cannot be extended, and it is potentially maximal. In this case, we add \( X_i \) to the set \( \mathcal{M} \), provided that \( X_i \) is not contained in any previously added maximal set \( Z \in \mathcal{M} \).
// Initial Call:  \( \mathcal{M} \leftarrow \emptyset, \)
\( P \leftarrow \{ \langle i, \text{t}(i) \rangle \mid i \in \mathcal{I}, \text{sup}(i) \geq \text{minsup} \} \)

**GenMax** (\( P, \text{minsup}, \mathcal{M} \)):

1. \( Y \leftarrow \bigcup X_i \)
2. \text{if } \exists Z \in \mathcal{M}, \text{such that } Y \subseteq Z \text{ then}
   \quad \text{return } // \text{prune entire branch}
3. \text{foreach } \langle X_i, \text{t}(X_i) \rangle \in P \text{ do}
   \quad P_i \leftarrow \emptyset
4. \quad \text{foreach } \langle X_j, \text{t}(X_j) \rangle \in P, \text{with } j > i \text{ do}
   \quad \quad X_{ij} \leftarrow X_i \cup X_j
   \quad \quad \text{t}(X_{ij}) = \text{t}(X_i) \cap \text{t}(X_j)
   \quad \quad \text{if } \text{sup}(X_{ij}) \geq \text{minsup} \text{ then } P_i \leftarrow P_i \cup \{ \langle X_{ij}, \text{t}(X_{ij}) \rangle \}\)
5. \quad \text{if } P_i \neq \emptyset \text{ then } \text{GenMax} (P_i, \text{minsup}, \mathcal{M})
6. \text{else if } \forall Z \in \mathcal{M}, X_i \subseteq Z \text{ then }
   \quad \mathcal{M} = \mathcal{M} \cup X_i // \text{add } X_i \text{ to maximal set}
Mining Maximal Frequent Itemsets

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1345</td>
<td>123456</td>
<td>2456</td>
<td>1356</td>
</tr>
</tbody>
</table>

P

PA

PA

PB

PC

PD

AB

AD

AE

1345

135

1345

BC

BD

BE

CE

DE

2456

1356

12345

245

135

12345

A

B

C

D

E

1345

135

1345

P

ABD

ABE

ADE

BCE

BDE

135

1345

135

245

135

135

P

ABDE

135
Mining closed frequent itemsets requires that we perform closure checks, that is, whether \( X = c(X) \). Direct closure checking can be very expensive. Given a collection of IT-pairs \( \{ \langle X_i, t(X_i) \rangle \} \), Charm uses the following three properties:

**Property (1)** If \( t(X_i) = t(X_j) \), then \( c(X_i) = c(X_j) = c(X_i \cup X_j) \), which implies that we can replace every occurrence of \( X_i \) with \( X_i \cup X_j \) and prune the branch under \( X_j \) because its closure is identical to the closure of \( X_i \cup X_j \).

**Property (2)** If \( t(X_i) \subset t(X_j) \), then \( c(X_i) \neq c(X_j) \) but \( c(X_i) = c(X_i \cup X_j) \), which means that we can replace every occurrence of \( X_i \) with \( X_i \cup X_j \), but we cannot prune \( X_j \) because it generates a different closure. Note that if \( t(X_i) \supset t(X_j) \) then we simply interchange the role of \( X_i \) and \( X_j \).

**Property (3)** If \( t(X_i) \neq t(X_j) \), then \( c(X_i) \neq c(X_j) \neq c(X_i \cup X_j) \). In this case we cannot remove either \( X_i \) or \( X_j \), as each of them generates a different closure.
Charm Algorithm: Closed Itemsets

// Initial Call:  \( C \leftarrow \emptyset \), \( P \leftarrow \{ \langle i, t(i) \rangle : i \in I, \text{sup}(i) \geq \text{minsup} \} \)

\textsc{Charm} (\( P \), \( \text{minsup} \), \( C \)):
1 Sort \( P \) in increasing order of support (i.e., by increasing \(|t(X_i)|\))
2 foreach \( \langle X_i, t(X_i) \rangle \in P \) do
3 \( P_i \leftarrow \emptyset \)
4 foreach \( \langle X_j, t(X_j) \rangle \in P \), with \( j > i \) do
5 \( X_{ij} = X_i \cup X_j \)
6 \( t(X_{ij}) = t(X_i) \cap t(X_j) \)
7 if \( \text{sup}(X_{ij}) \geq \text{minsup} \) then
8 \hfill if \( t(X_i) = t(X_j) \) then // Property 1
9 \hfill Replace \( X_i \) with \( X_{ij} \) in \( P \) and \( P_i \)
10 \hfill Remove \( \langle X_j, t(X_j) \rangle \) from \( P \)
11 \hfill else // Property 2
12 \hfill if \( t(X_i) \subset t(X_j) \) then // Property 3
13 \hfill Replace \( X_i \) with \( X_{ij} \) in \( P \) and \( P_i \)
14 \hfill else // Property 3
15 \hfill \( P_i \leftarrow P_i \cup \{ \langle X_{ij}, t(X_{ij}) \rangle \} \)
16 \hfill if \( P_i \neq \emptyset \) then \textsc{Charm} (\( P_i \), \( \text{minsup} \), \( C \))
17 \hfill if \( \forall Z \in C \), such that \( X_i \subseteq Z \) and \( t(X_i) = t(Z) \) then
18 \hfill \( C = C \cup X_i \) // Add \( X_i \) to closed set
Mining Frequent Closed Itemsets: Charm

Process A

\[ P_A \]

\[ \begin{array}{cccc}
A & AE & AEB & 1345 \\
C & 2456 & D & 1356 \\
E & 12345 & B & 123456
\end{array} \]

Min sup = 3
Mining Frequent Closed Itemsets: Charm

<table>
<thead>
<tr>
<th>A</th>
<th>A E</th>
<th>A E B</th>
<th>C</th>
<th>C B</th>
<th>D</th>
<th>D B</th>
<th>E</th>
<th>E B</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1345</td>
<td>2456</td>
<td>1356</td>
<td>12345</td>
<td>12345</td>
<td>123456</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ P_A, P_C, P_D \]

|   A D   |   A D E   |   A D E B   |   C E   |   C E B   |   D E   |   D E B   | 135  | 245    | 135  |

Zaki & Meira Jr. (RPI and UFMG)
Nonderivable Itemsets

An itemset is called *nonderivable* if its support cannot be deduced from the supports of its subsets. The set of all frequent nonderivable itemsets is a summary or condensed representation of the set of all frequent itemsets. Further, it is lossless with respect to support, that is, the exact support of all other frequent itemsets can be deduced from it.

**Generalized Itemsets:** Let $X$ be a $k$-itemset, that is, $X = \{x_1, x_2, \ldots, x_k\}$. The $k$ tidsets $t(x_i)$ for each item $x_i \in X$ induce a partitioning of the set of all tids into $2^k$ regions, where each partition contains the tids for some subset of items $Y \subseteq X$, but for none of the remaining items $Z = X \setminus Y$.

Each partition is therefore the tidset of a *generalized itemset* $Y\overline{Z}$, where $Y$ consists of regular items and $Z$ consists of negated items.

Define the support of a generalized itemset $Y\overline{Z}$ as the number of transactions that contain all items in $Y$ but no item in $Z$:

$$\text{sup}(Y\overline{Z}) = |\{t \in T \mid Y \subseteq i(t) \text{ and } Z \cap i(t) = \emptyset\}|$$
The inclusion–exclusion principle allows one to directly compute the support of $Y \overline{Z}$

$$\text{sup}(Y\overline{Z}) = \sum_{Y \subseteq W \subseteq X} -1^{|W\setminus Y|} \cdot \text{sup}(W)$$

From the $2^k$ possible subsets $Y \subseteq X$, we derive $2^{k-1}$ lower bounds and $2^{k-1}$ upper bounds for $\text{sup}(X)$, obtained after setting $\text{sup}(Y\overline{Z}) \geq 0$

**Upper Bounds** ($|X \setminus Y|$ is odd):  

$$\text{sup}(X) \leq \sum_{Y \subseteq W \subseteq X} -1^{(|X\setminus Y|+1)} \text{sup}(W)$$

**Lower Bounds** ($|X \setminus Y|$ is even):  

$$\text{sup}(X) \geq \sum_{Y \subseteq W \subseteq X} -1^{(|X\setminus Y|+1)} \text{sup}(W)$$
Tidset Partitioning Induced by $t(A), t(C),$ and $t(D)$

<table>
<thead>
<tr>
<th>Tid</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$ABDE$</td>
</tr>
<tr>
<td>2</td>
<td>$BCE$</td>
</tr>
<tr>
<td>3</td>
<td>$ABDE$</td>
</tr>
<tr>
<td>4</td>
<td>$ABCE$</td>
</tr>
<tr>
<td>5</td>
<td>$ABCDE$</td>
</tr>
<tr>
<td>6</td>
<td>$BCD$</td>
</tr>
</tbody>
</table>

$t(A) = 1356$
$t(C) = 2452$
$t(D) = 1352$

$\text{Sup}(A) \geq \text{Sup}(ACD) \geq \text{Sup}(AC) + \text{Sup}(AD) - \text{Sup}(A)$
Inclusion–Exclusion for Support

Consider the generalized itemset $\overline{ACD} = CAD$, where $Y = C$, $Z = AD$ and $X = YZ = ACD$. In the Venn diagram, we start with all the tids in $t(C)$, and remove the tids contained in $t(AC)$ and $t(CD)$. However, we realize that in terms of support this removes $\text{sup}(ACD)$ twice, so we need to add it back. In other words, the support of $CAD$ is given as

$$\text{sup}(CAD) = \text{sup}(C) - \text{sup}(AC) - \text{sup}(CD) + \text{sup}(ACD)$$

$$= 4 - 2 - 2 + 1 = 1$$

But, this is precisely what the inclusion–exclusion formula gives:

$$\text{sup}(CAD) = (-1)^0 \text{sup}(C) + (-1)^1 \text{sup}(AC) + (-1)^1 \text{sup}(CD) + (-1)^2 \text{sup}(ACD)$$

$$= \text{sup}(C) - \text{sup}(AC) - \text{sup}(CD) + \text{sup}(ACD)$$
Support Bounds

From each of the partitions, we get one bound, and out of the eight possible regions, exactly four give upper bounds and the other four give lower bounds for the support of \( ACD \):

\[
\begin{align*}
\sup(ACD) & \geq 0 \quad \text{when } Y = ACD \\
\sup(AC) & \leq \sup(ACD) \quad \text{when } Y = AC \\
\sup(AD) & \leq \sup(ACD) \quad \text{when } Y = AD \\
\sup(CD) & \leq \sup(ACD) \quad \text{when } Y = CD \\
\sup(AC) + \sup(AD) - \sup(A) & \geq \sup(ACD) \quad \text{when } Y = A \\
\sup(AC) + \sup(CD) - \sup(C) & \geq \sup(ACD) \quad \text{when } Y = C \\
\sup(AD) + \sup(CD) - \sup(D) & \geq \sup(ACD) \quad \text{when } Y = D \\
\sup(AC) + \sup(AD) + \sup(CD) - \sup(A) - \sup(C) - \sup(D) + \sup(\emptyset) & \leq \sup(ACD) \quad \text{when } Y = \emptyset 
\end{align*}
\]
Support Bounds for Subsets

The subset lattice

\[ l = x \]

\[ [l, u] \]

Then ACD is derivable

\[ l + u \leq ACD \text{ is non-derivable} \]

<table>
<thead>
<tr>
<th>sign</th>
<th>inequality</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \leq )</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>( \geq )</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>( \leq )</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ \sup(ACD) \geq \sup(AC) + \sup(AD) - \sup(A) \]
Nonderivable Itemsets

Given an itemset $X$, and $Y \subseteq X$, let $IE(Y)$ denote the summation

$$IE(Y) = \sum_{Y \subseteq W \subset X} -1^{(|X \setminus Y|+1)} \cdot sup(W)$$

Then, the sets of all upper and lower bounds for $sup(X)$ are given as

$$UB(X) = \left\{ IE(Y) \mid Y \subseteq X, |X \setminus Y| \text{ is odd} \right\}$$

$$LB(X) = \left\{ IE(Y) \mid Y \subseteq X, |X \setminus Y| \text{ is even} \right\}$$

An itemset $X$ is called nonderivable if $\max\{LB(X)\} \neq \min\{UB(X)\}$, which implies that the support of $X$ cannot be derived from the support values of its subsets; we know only the range of possible values, that is,

$$sup(X) \in \left[ \max\{LB(X)\}, \min\{UB(X)\} \right]$$

On the other hand, $X$ is derivable if $sup(X) = \max\{LB(X)\} = \min\{UB(X)\}$ because in this case $sup(X)$ can be derived exactly using the supports of its subsets. Thus, the set of all frequent nonderivable itemsets is given as

$$\mathcal{N} = \{ X \in \mathcal{F} \mid \max\{LB(X)\} \neq \min\{UB(X)\} \}$$

where $\mathcal{F}$ is the set of all frequent itemsets.
Nonderivable Itemsets: Example

Consider the support bound formulas for $\text{sup}(ACD)$. The lower bounds are

\[
\text{sup}(ACD) \geq 0 \\
\geq \text{sup}(AC) + \text{sup}(AD) - \text{sup}(A) = 2 + 3 - 4 = 1 \\
\geq \text{sup}(AC) + \text{sup}(CD) - \text{sup}(C) = 2 + 2 - 4 = 0 \\
\geq \text{sup}(AD) + \text{sup}(CD) - \text{sup}(D) = 3 + 2 - 4 = 0
\]

and the upper bounds are

\[
\text{sup}(ACD) \leq \text{sup}(AC) = 2 \\
\leq \text{sup}(AD) = 3 \\
\leq \text{sup}(CD) = 2 \\
\leq \text{sup}(AC) + \text{sup}(AD) + \text{sup}(CD) - \text{sup}(A) - \text{sup}(C) - \\
\text{sup}(D) + \text{sup}(\emptyset) = 2 + 3 + 2 - 4 - 4 - 4 + 6 = 1
\]

Thus, we have

$$\text{LB}(ACD) = \{0, 1\} \quad \text{max}\{\text{LB}(ACD)\} = 1$$

$$\text{UB}(ACD) = \{1, 2, 3\} \quad \text{min}\{\text{UB}(ACD)\} = 1$$

Because $\text{max}\{\text{LB}(ACD)\} = \text{min}\{\text{UB}(ACD)\}$ we conclude that $ACD$ is derivable.