

Data Mining and Machine Learning: Fundamental Concepts and Algorithms

dataminingbook.info

Mohammed J. Zaki¹ Wagner Meira Jr.²

¹Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

²Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 9: Summarizing Itemsets

An Example Database

Transaction database

Tid	Itemset
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Frequent itemsets ($minsup = 3$)

<i>sup</i>	Itemsets
6	B
5	E, BE
4	A, C, D, AB, AE, BC, BD, ABE
3	AD, CE, DE, ABD, ADE, BCE, BDE, ABDE

Maximal Frequent Itemsets

Given a binary database $\mathbf{D} \subseteq \mathcal{T} \times \mathcal{I}$, over the tids \mathcal{T} and items \mathcal{I} , let \mathcal{F} denote the set of all frequent itemsets, that is,

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

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

$$\mathcal{M} = \{X \mid X \in \mathcal{F} \text{ and } \nexists Y \supset X, \text{ such that } Y \in \mathcal{F}\}$$

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

Closed Frequent Itemsets

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

$$\mathcal{C} = \{X \mid X \in \mathcal{F} \text{ and } \nexists Y \supset X \text{ such that } \text{sup}(X) = \text{sup}(Y)\}$$

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 \mathcal{C} is a condensed representation, as we can determine whether an itemset X is frequent, as well as the exact support of X using \mathcal{C} alone.

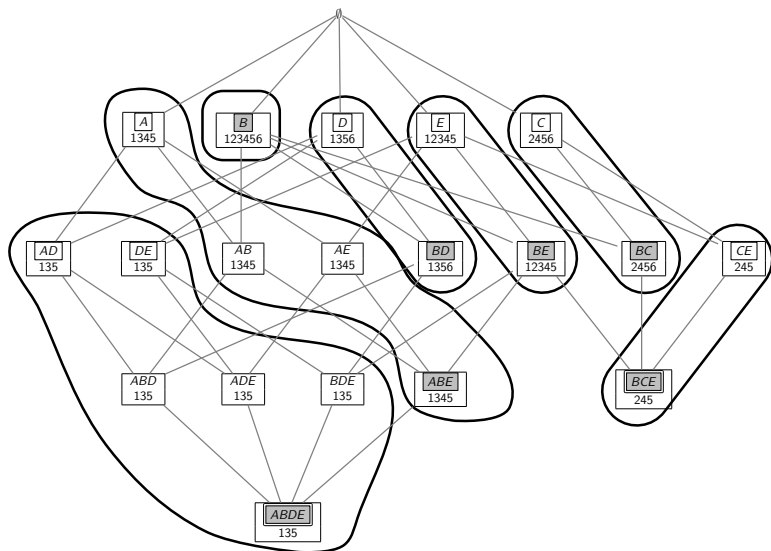
A frequent itemset X is a *minimal generator* if it has no subsets with the same support:

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

In other words, all subsets of X have strictly higher support, that is, $\text{sup}(X) < \text{sup}(Y)$, 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

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:** $\nexists Y \in \mathcal{M}$, 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:** $\nexists Y \in \mathcal{M}$, 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, $\langle i, \mathbf{t}(i) \rangle$, and the initially empty set of maximal itemsets, \mathcal{M} . Given a set of itemset–tidset pairs, called IT-pairs, of the form $\langle X, \mathbf{t}(X) \rangle$, the recursive GenMax method works as follows.

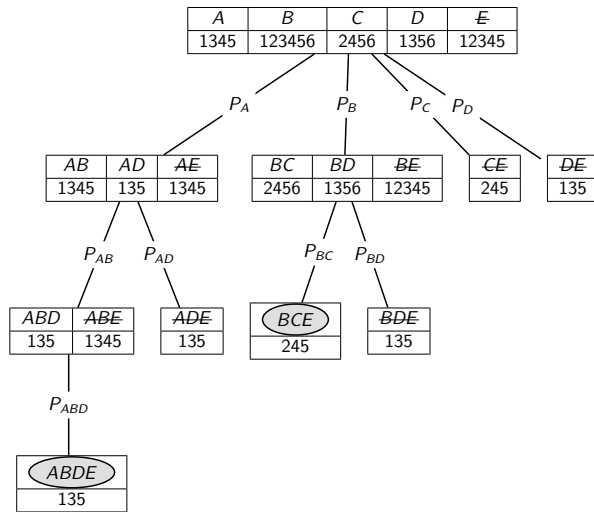
If the union of all the itemsets, $Y = \bigcup X_i$, is already subsumed by (or contained in) some maximal pattern $Z \in \mathcal{M}$, then no maximal itemset can be generated from the current branch, and it is pruned. Otherwise, we intersect each IT-pair $\langle X_i, \mathbf{t}(X_i) \rangle$ with all the other IT-pairs $\langle X_j, \mathbf{t}(X_j) \rangle$, 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 $Z \in \mathcal{M}$.

GenMax Algorithm

```
// Initial Call:  $\mathcal{M} \leftarrow \emptyset,$   
           $P \leftarrow \{ \langle i, \mathbf{t}(i) \rangle \mid i \in \mathcal{I}, \text{sup}(i) \geq \text{minsup} \}$   
GenMax ( $P, \text{minsup}, \mathcal{M}$ ):  
1  $Y \leftarrow \bigcup X_i$   
2 if  $\exists Z \in \mathcal{M}, \text{ such that } Y \subseteq Z$  then  
3    return // prune entire branch  
4 foreach  $\langle X_i, \mathbf{t}(X_i) \rangle \in P$  do  
5     $P_i \leftarrow \emptyset$   
6    foreach  $\langle X_j, \mathbf{t}(X_j) \rangle \in P, \text{ with } j > i$  do  
7       $X_{ij} \leftarrow X_i \cup X_j$   
8       $\mathbf{t}(X_{ij}) = \mathbf{t}(X_i) \cap \mathbf{t}(X_j)$   
9      if  $\text{sup}(X_{ij}) \geq \text{minsup}$  then  $P_i \leftarrow P_i \cup \{ \langle X_{ij}, \mathbf{t}(X_{ij}) \rangle \}$   
10    if  $P_i \neq \emptyset$  then GenMax ( $P_i, \text{minsup}, \mathcal{M}$ )  
11    else if  $\nexists Z \in \mathcal{M}, X_i \subseteq Z$  then  
12       $\mathcal{M} = \mathcal{M} \cup X_i$  // add  $X_i$  to maximal set
```

Mining Maximal Frequent Itemsets



Mining Closed Frequent Itemsets: Charm Algorithm

Mining closed frequent itemsets requires that we perform closure checks, that is, whether $X = \mathbf{c}(X)$. Direct closure checking can be very expensive.

Given a collection of IT-pairs $\{\langle X_i, \mathbf{t}(X_i) \rangle\}$, Charm uses the following three properties:

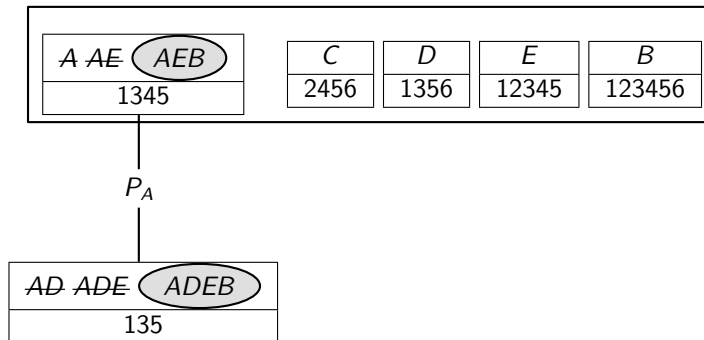
- If $\mathbf{t}(X_i) = \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) = \mathbf{c}(X_j) = \mathbf{c}(X_i \cup X_j)$, which implies that we can replace every occurrence of X_i with $X_i \cup X_j$ and prune the branch under X_j because its closure is identical to the closure of $X_i \cup X_j$.
- If $\mathbf{t}(X_i) \subset \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) \neq \mathbf{c}(X_j)$ but $\mathbf{c}(X_i) = \mathbf{c}(X_i \cup X_j)$, 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 $\mathbf{t}(X_i) \supset \mathbf{t}(X_j)$ then we simply interchange the role of X_i and X_j .
- If $\mathbf{t}(X_i) \neq \mathbf{t}(X_j)$, then $\mathbf{c}(X_i) \neq \mathbf{c}(X_j) \neq \mathbf{c}(X_i \cup X_j)$. In this case we cannot remove either X_i or X_j , as each of them generates a different closure.

Charm Algorithm: Closed Itemsets

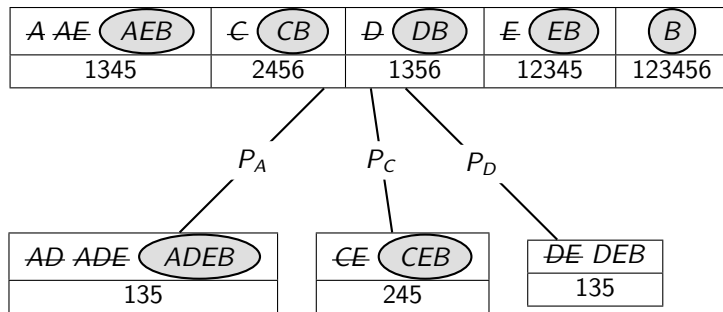
```
// Initial Call:  $C \leftarrow \emptyset$ ,  $P \leftarrow \{ \langle i, \mathbf{t}(i) \rangle : i \in \mathcal{I}, \text{sup}(i) \geq \text{minsup} \}$ 
Charm ( $P$ ,  $\text{minsup}$ ,  $C$ ):
1 Sort  $P$  in increasing order of support (i.e., by increasing  $|\mathbf{t}(X_i)|$ )
2 foreach  $\langle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

Mining Frequent Closed Itemsets: Charm

Process A



Mining Frequent Closed Itemsets: Charm



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, $X = \{x_1, x_2, \dots, x_k\}$. 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 $Z = X \setminus Y$.

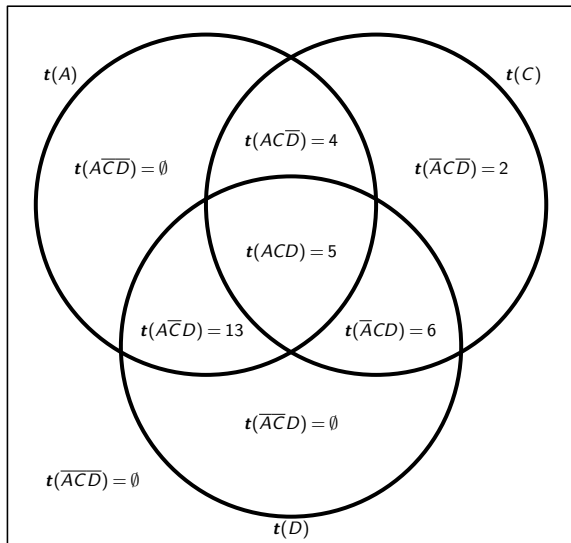
Each partition is therefore the tidset of a *generalized itemset* $Y\bar{Z}$, where Y consists of regular items and Z consists of negated items.

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

$$\text{sup}(Y\bar{Z}) = |\{t \in \mathcal{T} \mid Y \subseteq i(t) \text{ and } Z \cap i(t) = \emptyset\}|$$

Tidset Partitioning Induced by $t(A)$, $t(C)$, and $t(D)$

Tid	Itemset
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD



Inclusion–Exclusion Principle: Support Bounds

Let $Y\bar{Z}$ be a generalized itemset, and let $X = Y \cup Z = YZ$. The inclusion–exclusion principle allows one to directly compute the support of $Y\bar{Z}$ as a combination of the supports for all itemsets W , such that $Y \subseteq W \subseteq X$:

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

Inclusion–Exclusion for Support

Consider the generalized itemset $\overline{ACD} = \overline{CAD}$, where $Y = C$, $Z = AD$ and $X = YZ = ACD$. In the Venn diagram, we start with all the tids in $\mathbf{t}(C)$, and remove the tids contained in $\mathbf{t}(AC)$ and $\mathbf{t}(CD)$. 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 \overline{CAD} is given as

$$\begin{aligned}\text{sup}(\overline{CAD}) &= \text{sup}(C) - \text{sup}(AC) - \text{sup}(CD) + \text{sup}(ACD) \\ &= 4 - 2 - 2 + 1 = 1\end{aligned}$$

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

$$\begin{aligned}\text{sup}(\overline{CAD}) &= (-1)^0 \text{sup}(C) + && W = C, |W \setminus Y| = 0 \\ &(-1)^1 \text{sup}(AC) + && W = AC, |W \setminus Y| = 1 \\ &(-1)^1 \text{sup}(CD) + && W = CD, |W \setminus Y| = 1 \\ &(-1)^2 \text{sup}(ACD) && W = ACD, |W \setminus Y| = 2 \\ &= \text{sup}(C) - \text{sup}(AC) - && \\ &\text{sup}(CD) + \text{sup}(ACD)\end{aligned}$$

Support Bounds

Because the support of any (generalized) itemset must be non-negative, we can derive a bound on the support of X from each of the 2^k generalized itemsets by setting $\text{sup}(Y\bar{Z}) \geq 0$.

Thus, 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 $\text{sup}(Y\bar{Z}) \geq 0$:

$$\textbf{Upper Bounds (}|X \setminus Y| \text{ is odd): } \text{sup}(X) \leq \sum_{Y \subseteq W \subset X} -1^{(|X \setminus W|+1)} \text{sup}(W)$$

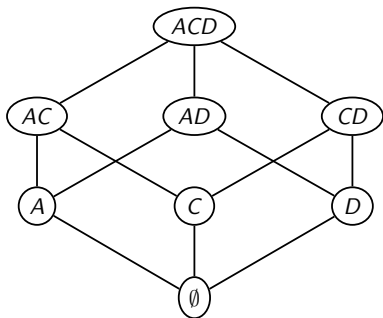
$$\textbf{Lower Bounds (}|X \setminus Y| \text{ is even): } \text{sup}(X) \geq \sum_{Y \subseteq W \subset X} -1^{(|X \setminus W|+1)} \text{sup}(W)$$

(1)

(2)

Support Bounds for Subsets

subset lattice



sign	inequality	level
1	\leq	1
-1	\geq	2
1	\leq	3

Support Bounds

Example

For example, if $Y = C$, then the inclusion-exclusion principle gives us

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

Setting $\text{sup}(C\overline{AD}) \geq 0$, we obtain

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

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 :

$\text{sup}(ACD) \geq 0$	when $Y = ACD$
$\leq \text{sup}(AC)$	when $Y = AC$
$\leq \text{sup}(AD)$	when $Y = AD$
$\leq \text{sup}(CD)$	when $Y = CD$
$\geq \text{sup}(AC) + \text{sup}(AD) - \text{sup}(A)$	when $Y = A$
$\geq \text{sup}(AC) + \text{sup}(CD) - \text{sup}(C)$	when $Y = C$
$\geq \text{sup}(AD) + \text{sup}(CD) - \text{sup}(D)$	when $Y = D$
$\leq \text{sup}(AC) + \text{sup}(AD) + \text{sup}(CD) -$ $\text{sup}(A) - \text{sup}(C) - \text{sup}(D) + \text{sup}(\emptyset)$	when $Y = \emptyset$

Nonderivable Itemsets

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

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

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

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

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

An itemset X is called *nonderivable* if $\max\{LB(X)\} \neq \min\{UB(X)\}$, and we know only the range of possible values, that is,

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

X is derivable if $sup(X) = \max\{LB(X)\} = \min\{UB(X)\}$. 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 $sup(ACD)$. The lower bounds are

$$\begin{aligned}sup(ACD) &\geq 0 \\ &\geq sup(AC) + sup(AD) - sup(A) = 2 + 3 - 4 = 1 \\ &\geq sup(AC) + sup(CD) - sup(C) = 2 + 2 - 4 = 0 \\ &\geq sup(AD) + sup(CD) - sup(D) = 3 + 2 - 4 = 0\end{aligned}$$

and the upper bounds are

$$\begin{aligned}sup(ACD) &\leq sup(AC) = 2 \\ &\leq sup(AD) = 3 \\ &\leq sup(CD) = 2 \\ &\leq sup(AC) + sup(AD) + sup(CD) - sup(A) - sup(C) - \\ &\quad sup(D) + sup(\emptyset) = 2 + 3 + 2 - 4 - 4 - 4 + 6 = 1\end{aligned}$$

Thus, we have

$$\begin{aligned}LB(ACD) &= \{0, 1\} & \max\{LB(ACD)\} &= 1 \\ UB(ACD) &= \{1, 2, 3\} & \min\{UB(ACD)\} &= 1\end{aligned}$$

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

Data Mining and Machine Learning: Fundamental Concepts and Algorithms

dataminingbook.info

Mohammed J. Zaki¹ Wagner Meira Jr.²

¹Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

²Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 9: Summarizing Itemsets