Data Mining and Machine Learning: Fundamental Concepts and Algorithms

datamininingbook.info

Mohammed J. Zaki[1] Wagner Meira Jr.[2]

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

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

Chapter 9: Summarizing Itemsets

Maximal Frequent Itemsets

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

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

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

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

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.

An Example Database

Transaction database

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

Frequent itemsets ($minsup = 3$)

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

Closed Frequent Itemsets

Given $T \subseteq \mathcal{T}$, and $X \subseteq \mathcal{I}$, define

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

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

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

X is closed if all supersets of X have strictly less support, that is, $sup(X) > sup(Y)$, 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.

Minimal Generators

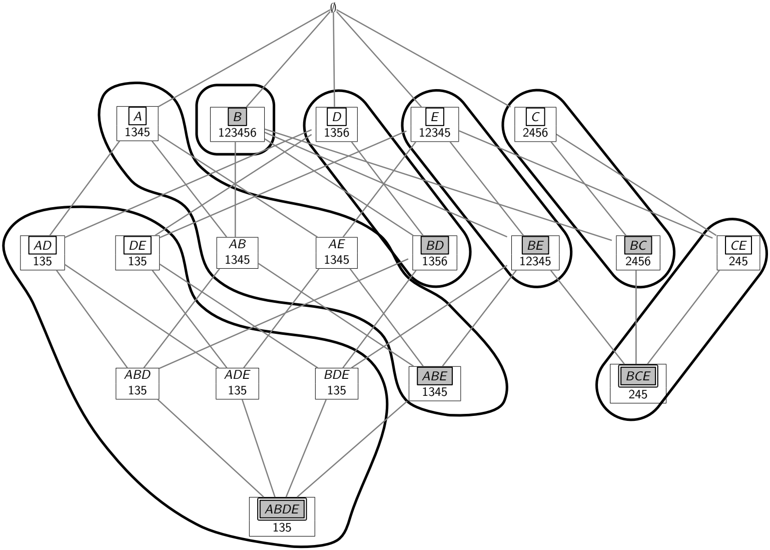
A frequent itemset X is a *minimal generator* if it has no subsets with the same support:

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

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

Mining Maximal Frequent Itemsets: GenMax Algorithm

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, $\langle i, \boldsymbol{t}(i) \rangle$, and the initially empty set of maximal itemsets, $\mathcal{M}$. Given a set of itemset–tidset pairs, called IT-pairs, of the form $\langle X, \boldsymbol{t}(X) \rangle$, 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 $\langle X_i, \boldsymbol{t}(X_i) \rangle$ with all the other IT-pairs $\langle X_j, \boldsymbol{t}(X_j) \rangle$, 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: $\mathcal{M} \leftarrow \emptyset$,

$P \leftarrow \{\langle i, \boldsymbol{t}(i) \rangle \mid i \in \mathcal{I}, sup(i) \geq minsup\}$

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

$Y \leftarrow \bigcup X_i$

if $\exists Z \in \mathcal{M}$, *such that* $Y \subseteq Z$ **then**

  return // prune entire branch

foreach $\langle X_i, \boldsymbol{t}(X_i) \rangle \in P$ **do**

  $P_i \leftarrow \emptyset$

  foreach $\langle X_j, \boldsymbol{t}(X_j) \rangle \in P$, *with* $j > i$ **do**

    $X_{ij} \leftarrow X_i \cup X_j$

    $\boldsymbol{t}(X_{ij}) = \boldsymbol{t}(X_i) \cap \boldsymbol{t}(X_j)$

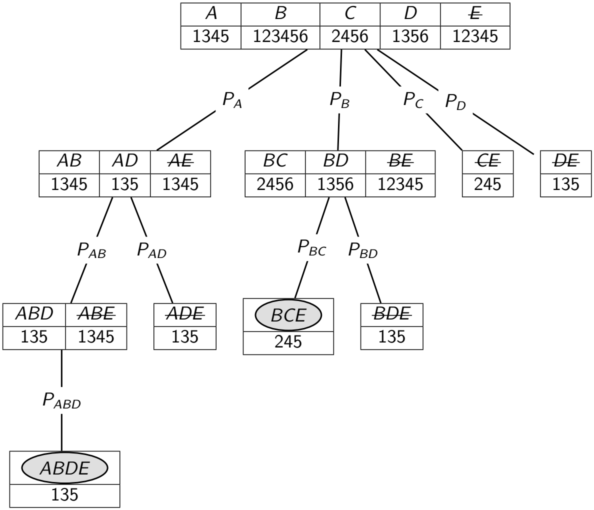
    if $sup(X_{ij}) \geq minsup$ **then** $P_i \leftarrow P_i \cup \{\langle X_{ij}, \boldsymbol{t}(X_{ij}) \rangle\}$

  if $P_i \neq \emptyset$ **then** GenMax (P_i, $minsup$, $\mathcal{M}$)

  else if $\nexists Z \in \mathcal{M}, X_i \subseteq Z$ **then**

    $\mathcal{M} = \mathcal{M} \cup X_i$ // 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 $X = \boldsymbol{c}(X)$. Direct closure checking can be very expensive.
Given a collection of IT-pairs $\{\langle X_i, \boldsymbol{t}(X_i)\rangle\}$, Charm uses the following three properties:

- If $\boldsymbol{t}(X_i) = \boldsymbol{t}(X_j)$, then $\boldsymbol{c}(X_i) = \boldsymbol{c}(X_j) = \boldsymbol{c}(X_i \cup X_j)$, 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 $\boldsymbol{t}(X_i) \subset \boldsymbol{t}(X_j)$, then $\boldsymbol{c}(X_i) \neq \boldsymbol{c}(X_j)$ but $\boldsymbol{c}(X_i) = \boldsymbol{c}(X_i \cup X_j)$, 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 $\boldsymbol{t}(X_i) \supset \boldsymbol{t}(X_j)$ then we simply interchange the role of X_i and X_j.
- If $\boldsymbol{t}(X_i) \neq \boldsymbol{t}(X_j)$, then $\boldsymbol{c}(X_i) \neq \boldsymbol{c}(X_j) \neq \boldsymbol{c}(X_i \cup X_j)$. 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: $\mathcal{C} \leftarrow \emptyset$, $P \leftarrow \{\langle i, \boldsymbol{t}(i)\rangle : i \in \mathcal{I}, sup(i) \geq minsup\}$

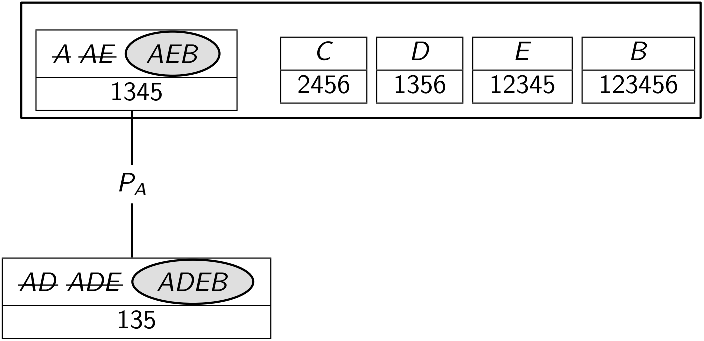
Charm (P, $minsup$, $\mathcal{C}$):

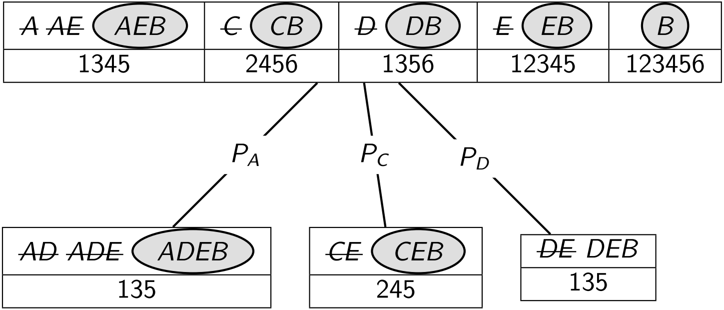
Sort P in increasing order of support (i.e., by increasing $|\boldsymbol{t}(X_i)|$)
foreach $\langle X_i, \boldsymbol{t}(X_i)\rangle \in P$ **do**
  $P_i \leftarrow \emptyset$
  foreach $\langle X_j, \boldsymbol{t}(X_j)\rangle \in P$, *with* $j > i$ **do**
    $X_{ij} = X_i \cup X_j$
    $\boldsymbol{t}(X_{ij}) = \boldsymbol{t}(X_i) \cap \boldsymbol{t}(X_j)$
    if $sup(X_{ij}) \geq minsup$ **then**
      if $\boldsymbol{t}(X_i) = \boldsymbol{t}(X_j)$ **then** // Property 1
        Replace X_i with X_{ij} in P and P_i
        Remove $\langle X_j, \boldsymbol{t}(X_j)\rangle$ from P
      else
        if $\boldsymbol{t}(X_i) \subset \boldsymbol{t}(X_j)$ **then** // Property 2
          Replace X_i with X_{ij} in P and P_i
        else // Property 3
          $P_i \leftarrow P_i \cup \{\langle X_{ij}, \boldsymbol{t}(X_{ij})\rangle\}$
  if $P_i \neq \emptyset$ **then** Charm (P_i, $minsup$, $\mathcal{C}$)

  if $\nexists Z \in \mathcal{C}$, *such that* $X_i \subseteq Z$ *and* $\boldsymbol{t}(X_i) = \boldsymbol{t}(Z)$ **then**
    $\mathcal{C} = \mathcal{C} \cup X_i$ // Add X_i to closed set

Mining Frequent Closed Itemsets: Charm

Process A




~~A~~ ~~AE~~ AEB
1345
~~C~~ CB
2456
~~D~~ DB
1356
~~E~~ EB
12345
B
123456
P_A
P_C
P_D
~~AD~~ ~~ADE~~ ADEB
135
~~CE~~ CEB
245
~~DE~~ DEB
135

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 $\boldsymbol{t}(x_i)$ 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:

$$\boxed{sup(Y\overline{Z}) = \left|\{t \in \mathcal{T} \mid Y \subseteq \boldsymbol{i}(t) \text{ and } Z \cap \boldsymbol{i}(t) = \emptyset\}\right|}$$

Inclusion–Exclusion Principle: Support Bounds

The inclusion–exclusion principle allows one to directly compute the support of $Y\overline{Z}$

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

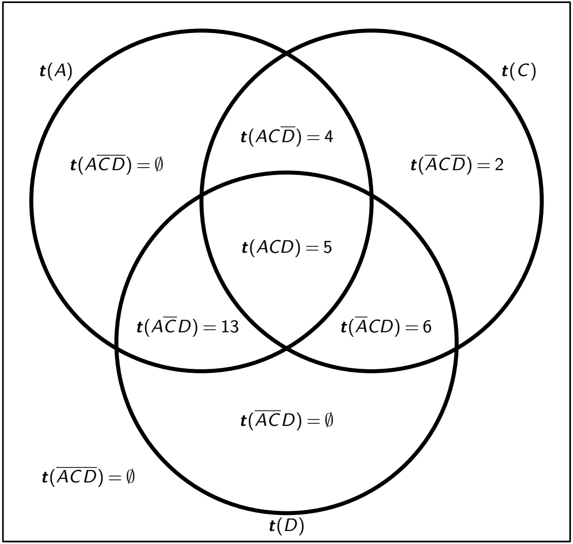
From the 2^k possible subsets $Y \subseteq X$, we derive 2^{k-1} lower bounds and 2^{k-1} upper bounds for $sup(X)$, obtained after setting $sup(Y\overline{Z}) \geq 0$

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

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

Tidset Partitioning Induced by $\boldsymbol{t}(A), \boldsymbol{t}(C)$, and $\boldsymbol{t}(D)$

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



Inclusion–Exclusion for Support

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

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

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

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

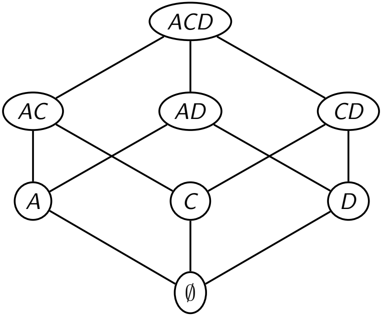
Support Bounds

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:

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

Support Bounds for Subsets

subset lattice



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

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

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

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

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,

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

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

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

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

datamininingbook.info

Mohammed J. Zaki[1]   Wagner Meira Jr.[2]

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

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

Chapter 9: Summarizing Itemsets