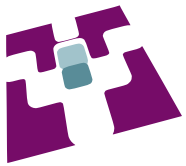
Itemset Mining: Fundamental Concepts and Algorithms

Samir Loudni

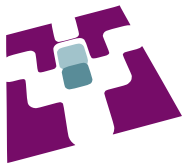
TASC (LS2N – CNRS) – DAPI
IMT Atlantique

FIL A3 – cours

29/01/2021



Outline



Data mining

... is "the use of sophisticated data analysis tools to **discover** unknown, **valid patterns and relationships** in large datasets"

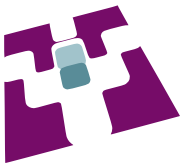
Data mining:

- Core of KDD
- Search for knowledge in data (regularities or correlations)
- Pattern domain : item-sets, sequences, graphs, etc.
- examples including pattern mining, clustering, association rules, etc.

	g_1	g_2	g_3	g_4
s_1	X			X
s_2	X	X	X	
s_3		X		X
s_4	X	X	X	
s_5	X	X		X

frequent pattern : $g_1 g_2$

association rule : $g_1 g_2 \rightarrow g_3$

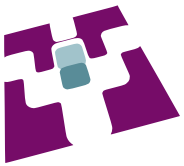


Frequent Itemset Mining : Motivation

In many applications one is interested in how often two or more objects of interest co-occur, the so-called itemsets.

The prototypical application was market basket analysis

- Finding regularities in the shopping behavior of customers of supermarkets, by analyzing the customer shopping carts (the so-called "market baskets").
  - **Find sets of products that are frequently bought together**.
- Possible applications of found frequent itemsets:
  - Improve arrangement of products in shelves, on a catalog's pages etc.
  - Support cross-selling (suggestion of other products), product bundling.
- Often found patterns are expressed as association rules, for example:
  If a customer buys **bread** and **wine**,
  then she/he will probably also buy **cheese**.

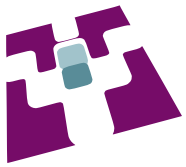


Frequent Itemsets: Terminology (1/2)

Itemsets: Let $\mathcal{I} = \{x_1, \ldots, x_n\}$ be a set of elements called items. A set $X \subseteq \mathcal{I}$ is called an itemset. An itemset of cardinality (or size) k is called a k-itemset.

Tidsets: Let $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ be another set of elements called transaction identifiers or tids. A set $T \subseteq \mathcal{T}$ is called a tidset.

Transactions: A transaction is a tuple of the form $\langle t, X \rangle$, where $t \in \mathcal{T}$ is a unique transaction identifier, and X is an itemset.



Frequent Itemsets: Terminology (2/2)

Database

A binary database $\mathcal{D}$ is a binary relation on the set of tids and items, that is, $\mathcal{D} \subseteq \mathcal{T} \times \mathcal{I}$. We say that tid $t \in \mathcal{T}$ contains item $x \in \mathcal{I}$ iff $(t, x) \in \mathcal{D}$. We say that tid t contains itemset $X = \{x_1, \ldots, x_k\}$ iff $(t, x_i) \in \mathcal{D}$ for all $i = 1, 2, \ldots, k$.

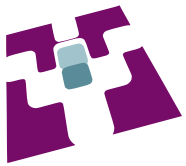
Input:

	x_1	x_2	$\ldots$	x_n
t_1	$d_{1,1}$	$d_{1,2}$	$\ldots$	$d_{1,n}$
t_2	$d_{2,1}$	$d_{2,2}$	$\ldots$	$d_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_m	$d_{m,1}$	$d_{m,2}$	$\ldots$	$d_{m,n}$

where $d_{i,j} \in \{\text{true,false}\}$

Size of the space search

How many itemsets are there ? $2^{|\mathcal{I}|}$.



Transactional representation of the data

Relational representation: $\mathcal{D} \subseteq \mathcal{T} \times \mathcal{I}$

	x_1	x_2	$\ldots$	x_n
t_1	$d_{1,1}$	$d_{1,2}$	$\ldots$	$d_{1,n}$
t_2	$d_{2,1}$	$d_{2,2}$	$\ldots$	$d_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_m	$d_{m,1}$	$d_{m,2}$	$\ldots$	$d_{m,n}$

where $d_{i,j} \in \{\text{true,false}\}$

Transactional representation: $\mathcal{D}$ is a collection of tuples of the form $\langle t, \boldsymbol{i}(t) \rangle$, with $t \in \mathcal{T}$

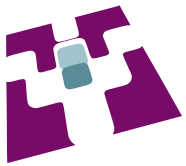
$$\langle t_1, \boldsymbol{i}(t_1) \rangle$$
$$\langle t_2, \boldsymbol{i}(t_2) \rangle$$
$$\vdots$$
$$\langle t_m, \boldsymbol{i}(t_m) \rangle$$

where $\boldsymbol{i}(t)$ is the set of items contained in tid $t \in \mathcal{T}$.

Example

	x_1	x_2	x_3
t_1	$\times$	$\times$	$\times$
t_2	$\times$	$\times$	
t_3		$\times$	
t_4			$\times$

t_1	x_1, x_2, x_3
t_2	x_1, x_2
t_3	x_2
t_4	x_3



Example of Database

$\mathcal{D}$	A	B	C	D	E
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

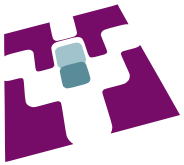
Binary Database

t	$\boldsymbol{i}(t)$
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Trans. Data.

This dataset $\mathcal{D}$ has 5 items, $\mathcal{I} = \{A, B, C, D, E\}$ and 6 tids $\mathcal{T} = \{1, 2, 3, 4, 5, 6\}$.

The first transaction is $\langle 1, \{A, B, D, E\}\rangle$, where we omit item C since $(1, C) \notin \mathcal{D}$. For convenience, we write $\langle 1, \{A, B, D, E\}\rangle$ as $\langle 1, ABDE\rangle$.



Frequency: definition

The frequency of an itemset X in a dataset $\mathcal{D}$, denoted $freq(X)$, is the number of transactions in $\mathcal{D}$ that contain X:

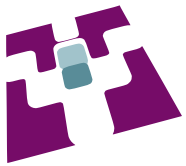
$$freq(X) = \left|\{t \mid \langle t, \boldsymbol{i}(t)\rangle \in \mathcal{D} \text{ and } X \subseteq \boldsymbol{i}(t)\}\right|$$

The relative frequency of X is the fraction of transactions that contain X:

$$rfreq(X) = \frac{sup(X)}{|\mathcal{D}|}$$

The relative frequency is a joint probability of the items comprising X.

An itemset X is said to be frequent in $\mathcal{D}$ if $freq(X) \geq \theta$, where θ is a user defined **minimum support threshold**.



Frequent Itemset mining

Problem Definition

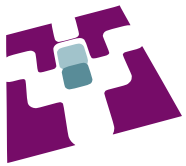
Given the tids in $\mathcal{T}$ described with the Boolean attributes in $\mathcal{I}$, listing every itemset having a frequency above a given threshold $\theta \in \mathbb{N}$.

Input:

	x_1	x_2	$\dots$	x_n
t_1	$d_{1,1}$	$d_{1,2}$	$\dots$	$d_{1,n}$
t_2	$d_{2,1}$	$d_{2,2}$	$\dots$	$d_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_m	$d_{m,1}$	$d_{m,2}$	$\dots$	$d_{m,n}$

where $d_{i,j} \in \{\text{true,false}\}$

and a minimal frequency $\theta \in \mathbb{N}$.

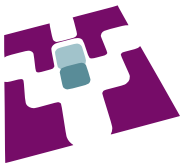


Frequent Itemset mining

Problem Definition

Given the tids in $\mathcal{T}$ described with the Boolean attributes in $\mathcal{I}$, listing every itemset having a frequency above a given threshold $\theta \in \mathbb{N}$.

Output: every $X \subseteq \mathcal{I}$ such that there are at least θ objects having all attributes in X.



Frequent Itemset mining: illustration
Minimum support: $\theta = 3$

t	$\boldsymbol{i}(t)$
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

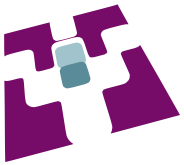
Transaction Database

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

Frequent Itemsets

The 19 frequent itemsets shown in the table comprise the set $\mathcal{F}$. The sets of all frequent k-itemsets are

$$\mathcal{F}^{(1)} = \{A, B, C, D, E\}$$
$$\mathcal{F}^{(2)} = \{AB, AD, AE, BC, BD, BE, CE, DE\}$$
$$\mathcal{F}^{(3)} = \{ABD, ABE, ADE, BCE, BDE\}$$
$$\mathcal{F}^{(4)} = \{ABDE\}$$

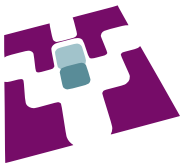


Itemset Mining Algorithms: Brute Force

The brute-force algorithm enumerates all the possible itemsets $X \subseteq \mathcal{I}$, and for each such subset determines its support in the input dataset $\mathcal{D}$. The method comprises two main steps:

Candidate Generation: This step generates all the subsets of $\mathcal{I}$, which are called candidates, as each itemset is potentially a candidate frequent pattern.

Frequency Computation: This step computes the support of each candidate pattern X and determines if it is frequent. For each transaction $\langle t, \boldsymbol{i}(t) \rangle$ in the database, we test if X is a subset of $\boldsymbol{i}(t)$. If so, we increment the frequency of X.



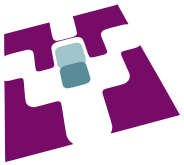
Brute Force Algorithm

BruteForce ($\mathcal{D}$, $\mathcal{I}$, θ)**:**

```
1 F ← ∅ // set of frequent itemsets
2
3 foreach X ⊆ I do
4     freq(X) ← ComputeSupport (X, D)
5     if freq(X) ≥ θ then
6         F ← F ∪ {(X, freq(X))}
7 return F
```

ComputeSupport ($X, \mathcal{D}$)**:**

```
1 sup(X) ← 0
2 foreach ⟨t, i(t)⟩ ∈ D do
3     if X ⊆ i(t) then
4         freq(X) ← freq(X) + 1
5 return freq(X)
```

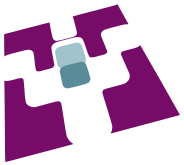


Complexity of the naive approach

Question
How many itemsets are enumerated ?

Question
What is the worst-case complexity of the frequency computation step ?

Question
What is the worst-case complexity of the naive approach?

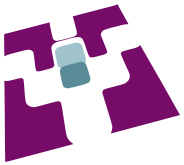


Complexity of the naive approach

Question
How many itemsets are enumerated ? $2^{|\mathcal{I}|}$

Question
What is the worst-case complexity of the frequency computation step ?

Question
What is the worst-case complexity of the naive approach?

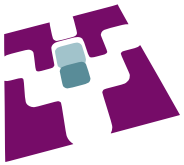


Complexity of the naive approach

Question
How many itemsets are enumerated ? $2^{|\mathcal{I}|}$

Question
What is the worst-case complexity of the frequency computation step ?
Support computation takes time $O(|\mathcal{I}| \cdot |\mathcal{D}|)$ in the worst case.

Question
What is the worst-case complexity of the naive approach?

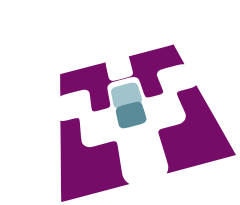


Complexity of the naive approach

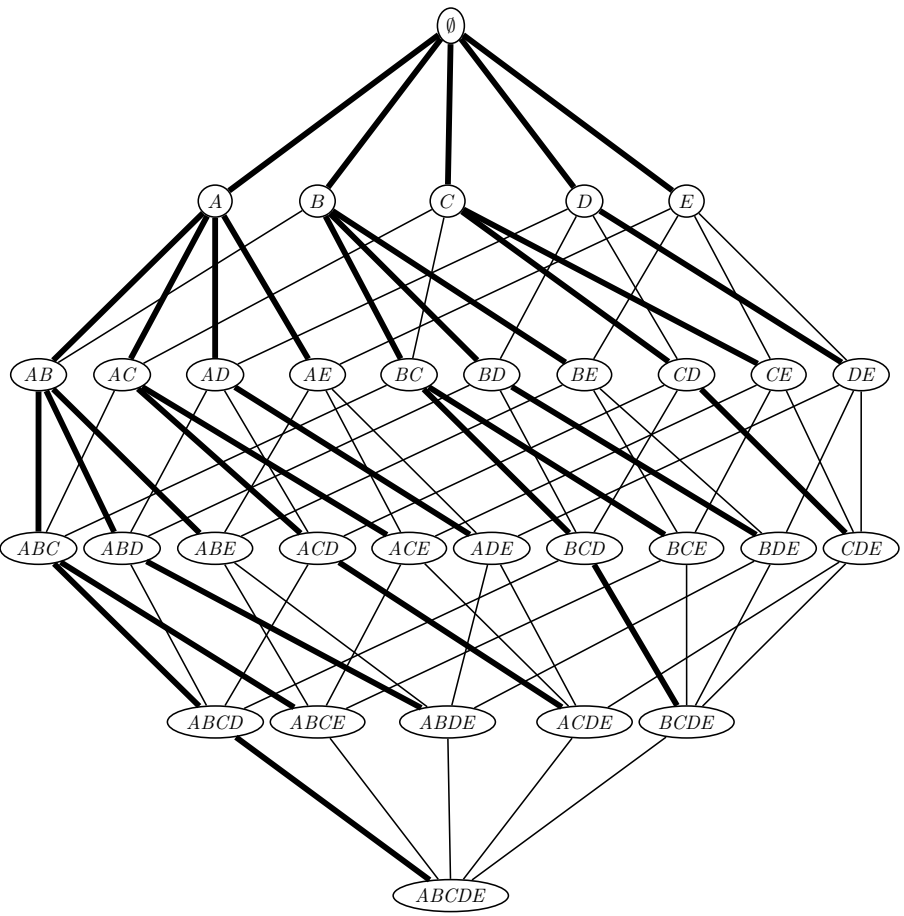
Question
How many itemsets are enumerated ? $2^{|\mathcal{I}|}$

Question
What is the worst-case complexity of the frequency computation step ?
Support computation takes time $O(|\mathcal{I}| \cdot |\mathcal{D}|)$ in the worst case.

Question
What is the worst-case complexity of the naive approach?
As there are $O(2^{|\mathcal{I}|})$ possible candidates, the overall computational complexity of the brute-force method is $O(|\mathcal{I}| \cdot |\mathcal{D}| \cdot 2^{|\mathcal{I}|})$.



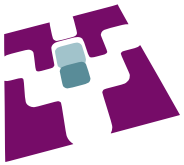
Itemset lattice and prefix-based enumeration



Itemset search space is a lattice where any two itemsets X and Y are connected by a link iff X is an immediate subset of Y, that is, $X \subseteq Y$ and $|X| = |Y| - 1$.

Frequent itemsets can enumerated using DFS search on the prefix tree. This allows one to mine itemsets starting with an empty set, and adding one more item at a time.

How to efficiently mine frequent itemsets?



Properties of the Support of Itemsets (1/2)

A **brute force approach** that traverses all possible item sets, determines their support, and discards infrequent item sets is usually **infeasible**:

Idea: Consider the properties of an itemset's cover and support, in particular:

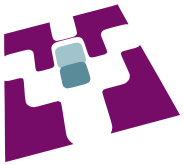
$$\forall X : \forall Y \supseteq X : \quad cover(Y) \subseteq cover(X).$$

It follows:

$$\forall X : \forall Y \supseteq X : \quad freq(Y, \mathcal{D}) < freq(X, \mathcal{D})$$

That is: **If an itemset is extended, its support cannot increase.**

One also says that support is **anti-monotone** or **downward closed**.



Properties of the Support of Itemsets (2/2)

From $\forall X : \forall Y \supseteq X : \quad freq(Y, \mathcal{D}) < freq(X, \mathcal{D})$ it follows immediately

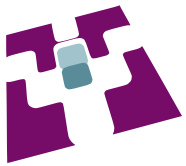
$$\forall \theta : \forall X : \forall Y \supseteq X : \quad freq(X, \mathcal{D}) < \theta \;\Rightarrow\; freq(Y, \mathcal{D}) < \theta$$

That is: **No superset of an infrequent itemset can be frequent.**

Of course, the contraposition of this implication also holds:

$$\forall \theta : \forall X : \forall Y \subseteq X : \quad freq(X, \mathcal{D}) \geq \theta \;\Rightarrow\; freq(Y, \mathcal{D}) \geq \theta$$

That is: **All subsets of a frequent itemset are frequent.**



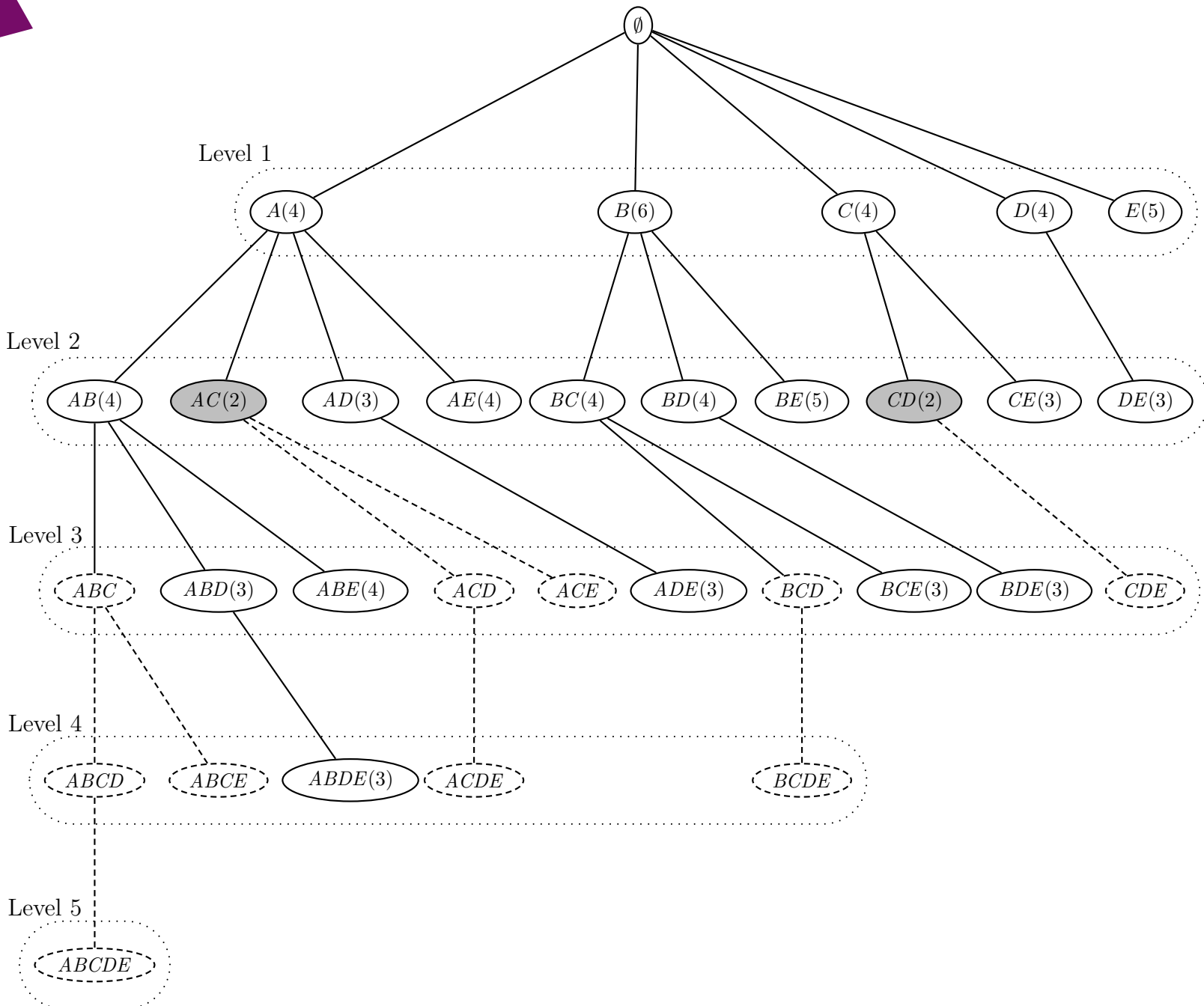
Level-wise Approach: Apriori Algorithm [Agrawal and Srikant 1994]

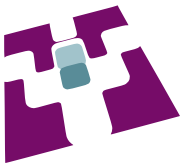
The **Apriori algorithm** utilizes the anti-monotonicity of the frequency to significantly improve the brute-force approach.

A level-wise or breadth-first exploration of the itemset search space to prunes all supersets of any infrequent candidate.
➠ Improving the candidate generation step

The **Apriori method** also significantly improves the I/O complexity. Instead of counting the support for a single itemset, it computes the support of all the valid candidates of size k that comprise level k in the prefix tree.

Level-wise enumeration of the itemsets

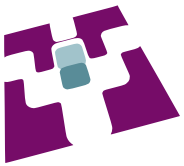




The Apriori Algorithm

Apriori ($\mathcal{D}$, $\mathcal{I}$, θ)**:**

1 $\mathcal{F} \leftarrow \emptyset$
2 $\mathcal{C}^{(1)} \leftarrow \{\emptyset\}$ // Initial prefix tree with single items
3
4 **foreach** $i \in \mathcal{I}$ **do** Add i as child of $\emptyset$ in $\mathcal{C}^{(1)}$ with $sup(i) \leftarrow 0$
5
6 $k \leftarrow 1$ // k denotes the level
7 **while** $\mathcal{C}^{(k)} \neq \emptyset$ **do**
8 ComputeSupport $(\mathcal{C}^{(k)}, \mathcal{D})$
9 **foreach** $X \in \mathcal{C}^{(k)}$ **do**
10 **if** $sup(X) \geq \theta$ **then** $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$
11 **else** remove X from $\mathcal{C}^{(k)}$
12 $\mathcal{C}^{(k+1)} \leftarrow$ ExtendPrefixTree $(\mathcal{C}^{(k)})$
13 $k \leftarrow k + 1$
14 **return** $\mathcal{F}^{(k)}$



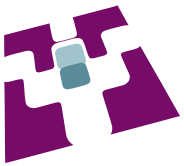
Apriori Algorithm

ComputeSupport ($\mathcal{C}^{(k)}, \mathcal{D}$)**:**

foreach $\langle t, \boldsymbol{i}(t) \rangle \in \mathcal{D}$ **do**
  foreach k-subset $X \subseteq \boldsymbol{i}(t)$ **do**
    if $X \in \mathcal{C}^{(k)}$ **then** $sup(X) \leftarrow sup(X) + 1$

ExtendPrefixTree ($\mathcal{C}^{(k)}$)**:**

foreach $X_a \in \mathcal{C}^{(k)}$ **do**
  foreach X_b *having common parent with* $X_a \wedge b > a$ **do**
    $X_{ab} \leftarrow X_a \cup X_b$
    `// prune candidate if there are any infrequent subsets`
    if $X_j \in \mathcal{C}^{(k)}$, **for all** $X_j \subset X_{ab}$, such that $|X_j| = |X_{ab}| - 1$ **then**
      Add X_{ab} as child of X_a with $sup(X_{ab}) \leftarrow 0$
  if no extensions from X_a **then**
    remove X_a, and all ancestors of X_a with no extensions, from $\mathcal{C}^{(k)}$
return $\mathcal{C}^{(k)}$



Apriori Algorithm: Details

Apriori begins by inserting the single items into an initially empty prefix tree to populate $\mathcal{C}^{(1)}$.

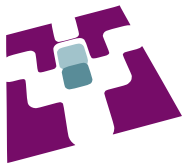
The itemsets are checked in the **order of increasing size** (**breadth-first / levelwise traversal**).

The support for the current candidates is obtained via ComputeSupport procedure. Next, we remove any infrequent candidate.

The set of frequent k-itemsets $\mathcal{F}^{(k)}$ are used to generate the candidate $(k+1)$-itemsets for the next level (using the **apriori property**).

The ExtendPrefixTree procedure employs prefix-based extension for candidate generation. A candidate of $(k+1)$-length is retained only if it has no infrequent subset.

If new candidates were added, the whole process is repeated for the next level. This process continues until no new candidates are added.

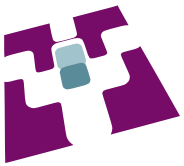


Apriori : Levelwise Search

t	$\boldsymbol{i}(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE

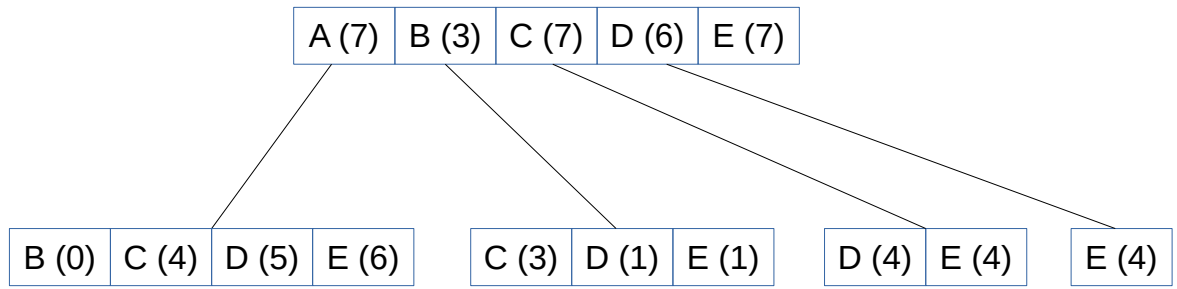
A (7)	B (3)	C (7)	D (6)	E (7)

- Example transaction database with 5 items and 10 transactions.
- Minimum support: 30%, that is, at least 3 transactions must contain the itemset.
- All sets with one item (singletons) are frequent $\Rightarrow$ full second level is needed.

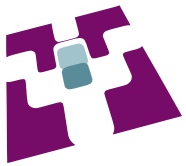


Apriori : Levelwise Search

t	$\boldsymbol{i}(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE

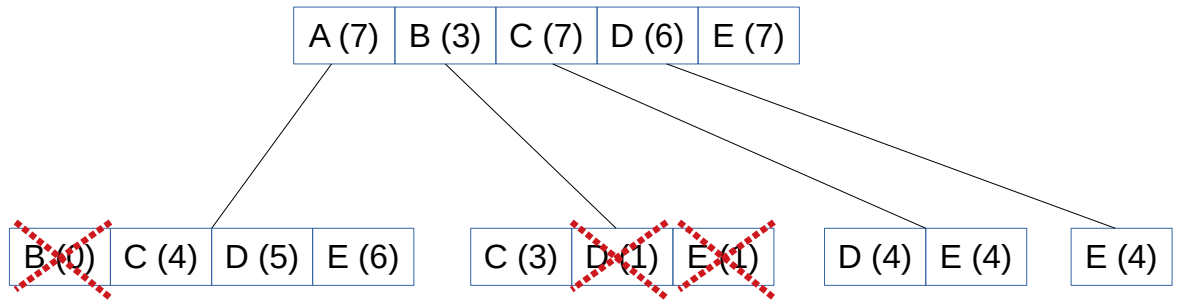


- Determining the support of itemsets: For each itemset traverse the database and count the transactions that contain it (highly inefficient).
- Better: Traverse the tree for each transaction and find the item sets it contains (efficient: can be implemented as a simple recursive procedure).

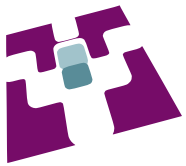


Apriori : Levelwise Search

t	$i(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE

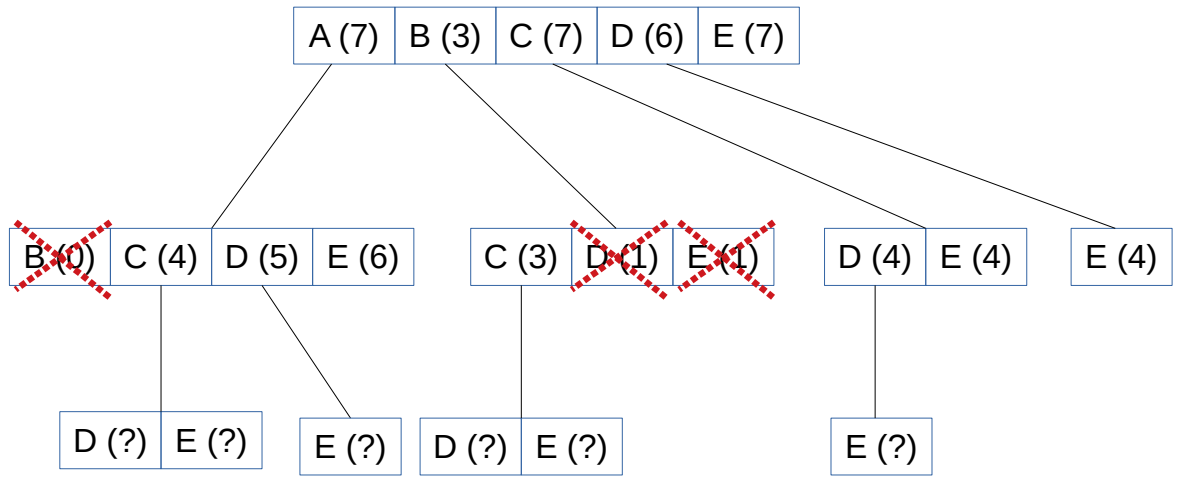


- Minimum support: 30%, that is, at least 3 transactions must contain the itemset.
- Infrequent itemsets: $\{a, b\}$, $\{b, d\}$, $\{b, e\}$.
- The subtrees starting at these itemsets can be pruned.
  (*a posteriori*: after accessing the transaction database to determine the support)

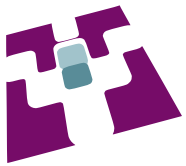


Apriori : Levelwise Search

t	$\boldsymbol{i}(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE

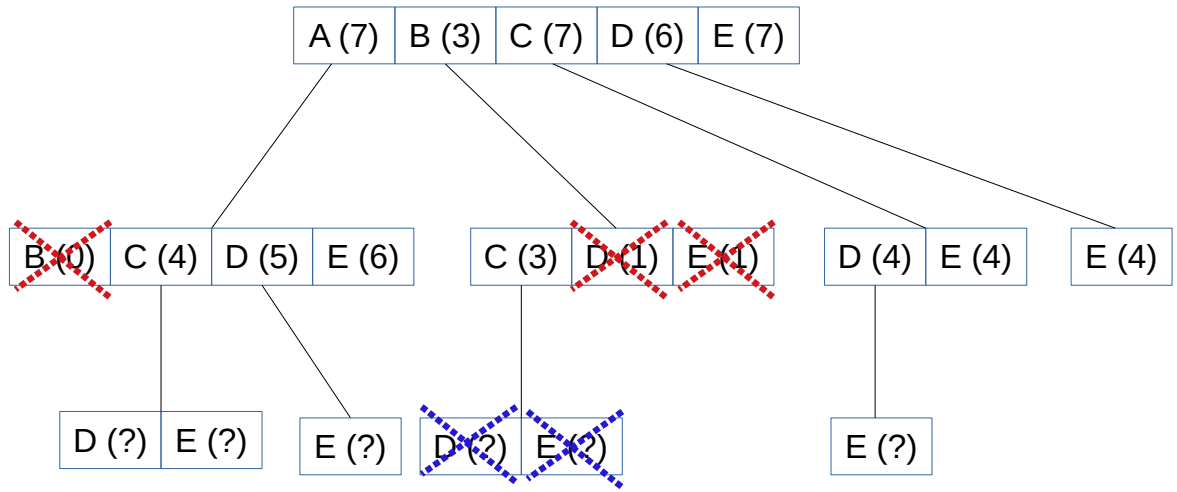


- Generate candidate itemsets with 3 items (parents must be frequent).
- Before counting, check whether the candidates contain an infrequent itemset.
  - An itemset with k items has k subsets of size $k-1$.
  - The parent itemset is only one of these subsets.

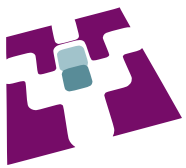


Apriori : Levelwise Search

t	$i(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE

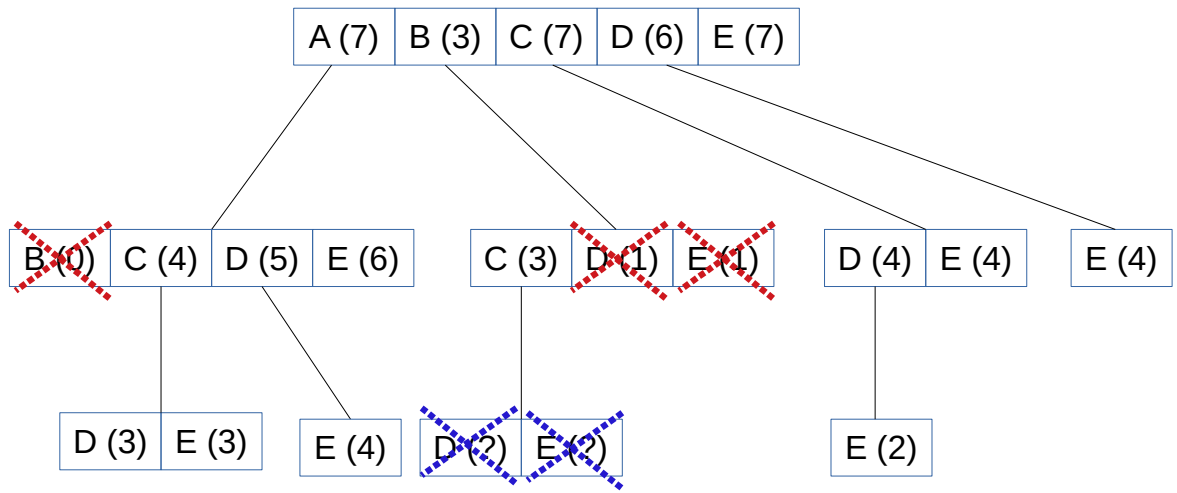


- The itemsets $\{b, c, d\}$ and $\{b, c, e\}$ can be pruned, because
  - $\{b, c, d\}$ contains the infrequent itemset $\{b, d\}$ and
  - $\{b, c, e\}$ contains the infrequent itemset $\{b, e\}$.
- *a priori*: before accessing the transaction database to determine the support



Apriori : Levelwise Search

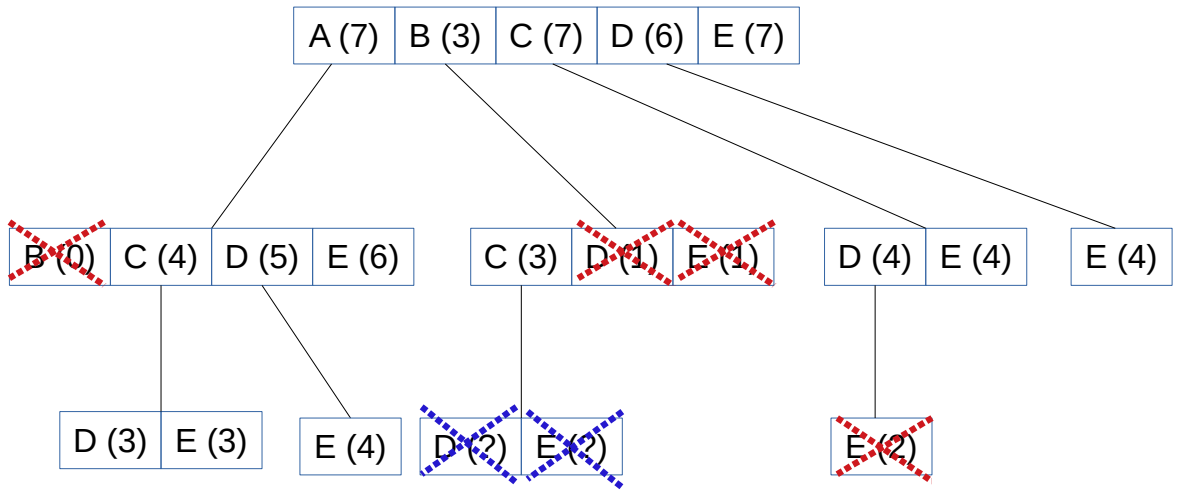
t	$\boldsymbol{i}(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE



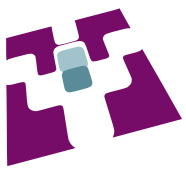
- Only the remaining four itemsets of size 3 are evaluated.
- No other itemsets of size 3 can be frequent.

Apriori : Levelwise Search

t	$i(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE

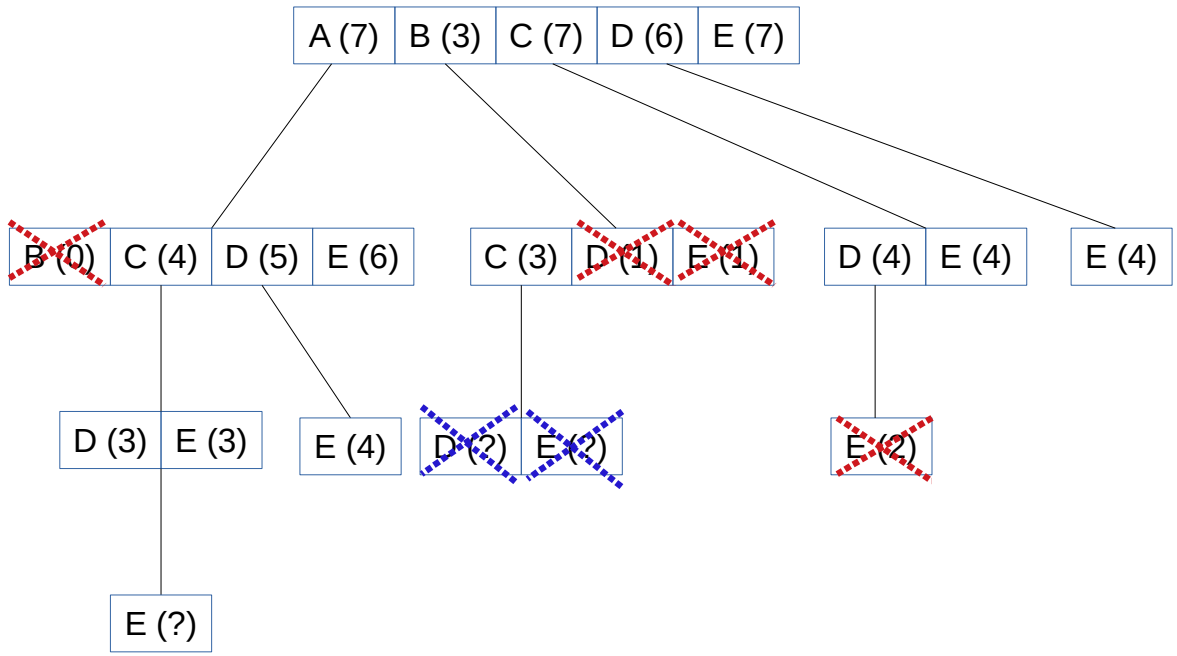


- Minimum support: 30%, that is, at least 3 transactions must contain the itemset.
- The infrequent itemset $\{c, d, e\}$ is pruned.
  (*a posteriori*: after accessing the transaction database to determine the support)

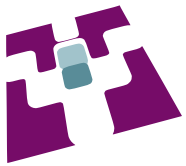


Apriori : Levelwise Search

t	$i(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE

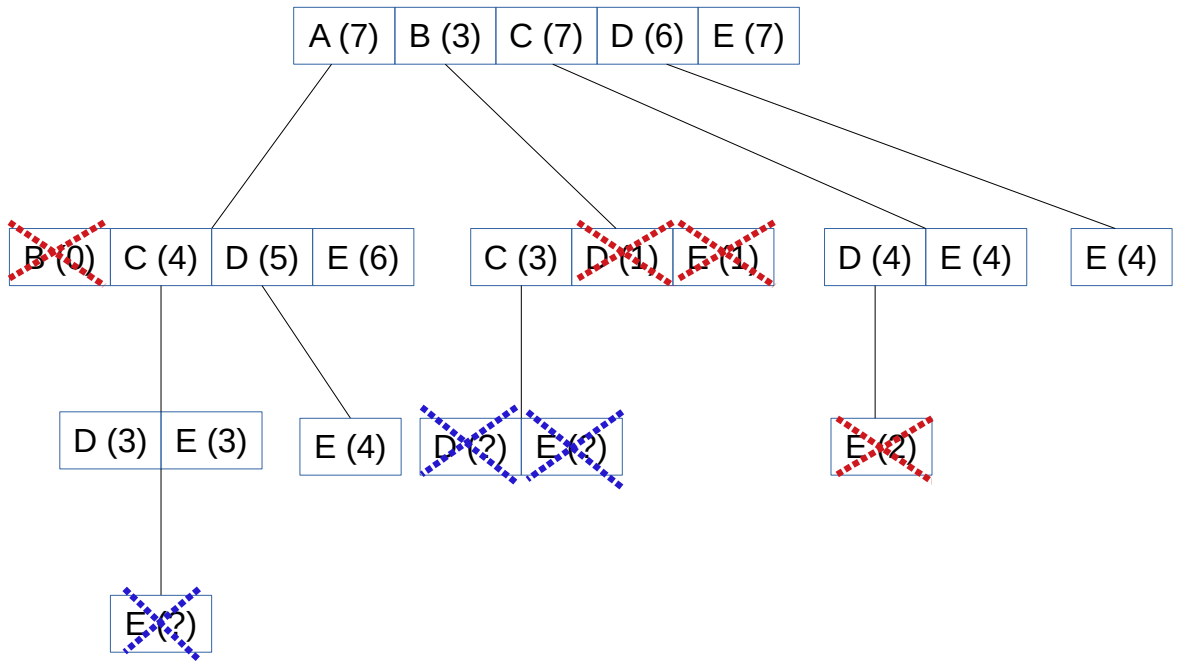


- Generate candidate itemsets with 4 items (parents must be frequent).
- Before counting, check whether the candidates contain an infrequent itemset.

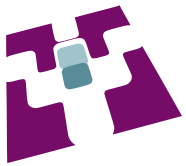


Apriori : Levelwise Search

t	$i(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE



- The item set $\{a, c, d, e\}$ can be pruned, because it contains the infrequent item set $\{c, d, e\}$.
- Consequence: No candidate itemsets with four items.
- Fourth access to the transaction database is not necessary.

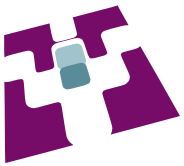


Transaction Database Representation

- The Apriori algorithm uses a **horizontal transaction representation**: each transaction is an array of the contained items.

- The alternative is a **vertical transaction representation**:
  - For each item a **transaction list** is created.
  - The transaction list of item x indicates the transactions that contain it, that is, it represents its **cover** $\mathcal{V}_{\mathcal{D}}(\{x\})$.
  - Advantage: the transaction list for a pair of items can be computed by intersecting the transaction lists of the individual items.
  - Generally, a vertical transaction representation can exploit

$$\forall X, Y \subseteq \mathcal{I}: \quad \mathcal{V}_{\mathcal{D}}(X \cup Y) = \mathcal{V}_{\mathcal{D}}(X) \cap \mathcal{V}_{\mathcal{D}}(Y).$$



Transaction Database Representation

$\mathcal{D}$	A	B	C	D	E
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

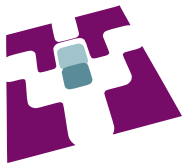
Binary Database

t	$\boldsymbol{i}(t)$
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Tran. Data.

$\boldsymbol{t}(x)$				
A	B	C	D	E
1	1	2	1	1
3	2	4	3	2
4	3	5	5	3
5	4	6	6	4
	5			5
	6			

Vertical Data.



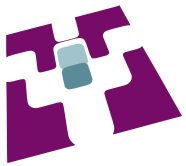
Depth-First Search and Conditional Databases

- A depth-first search can also be used to explore the search space of itemsets
  ➠ **divide-and-conquer scheme**

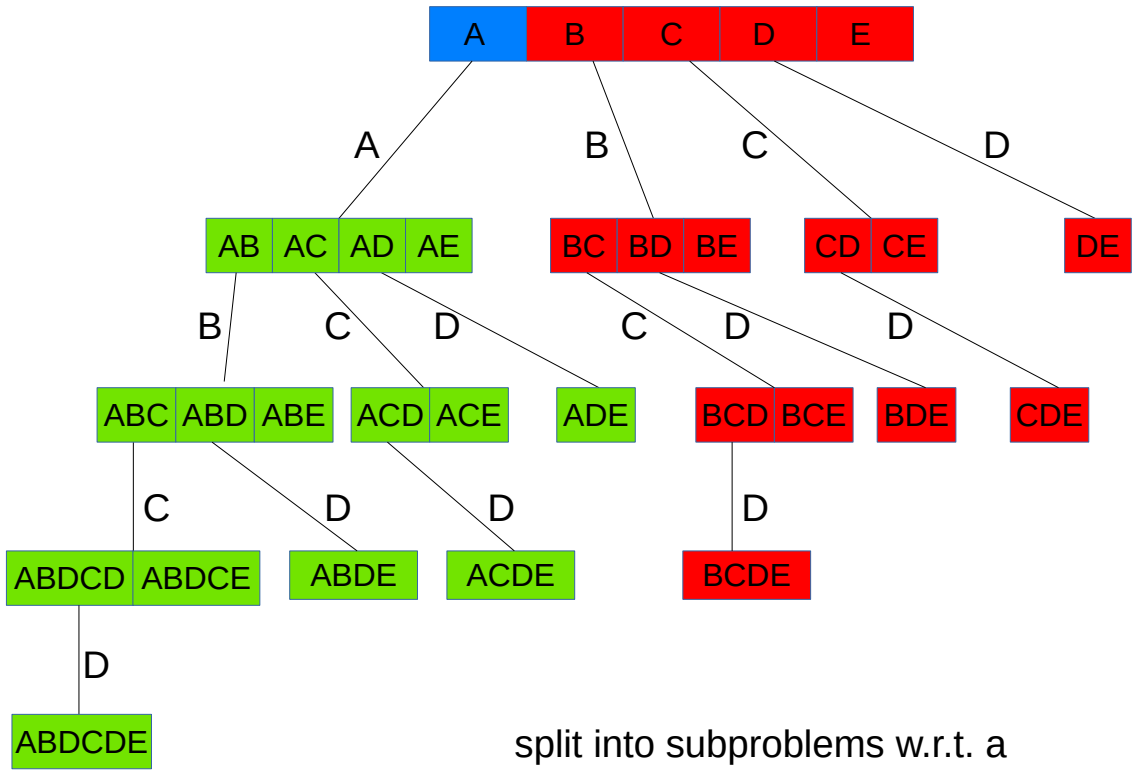
  First find all frequent itemsets that contain a chosen item, then all frequent itemsets that do not contain it.

- General search procedure: Let the item order be $a < b < c < \cdots$.
  - Restrict the transaction database to those transactions that contain a.
    ➠ **conditional database for the prefix** a.
  - Recursively search this conditional database for frequent itemsets and add the prefix a to all frequent itemsets found in the recursion.
  - Remove the item a from the transactions in the *full* transaction database.
    ➠ **conditional database for itemsets without** a.
    Recursively search this conditional database for frequent itemsets.

- With this scheme only frequent one-element itemsets have to be determined. Larger itemsets result from adding possible prefixes.

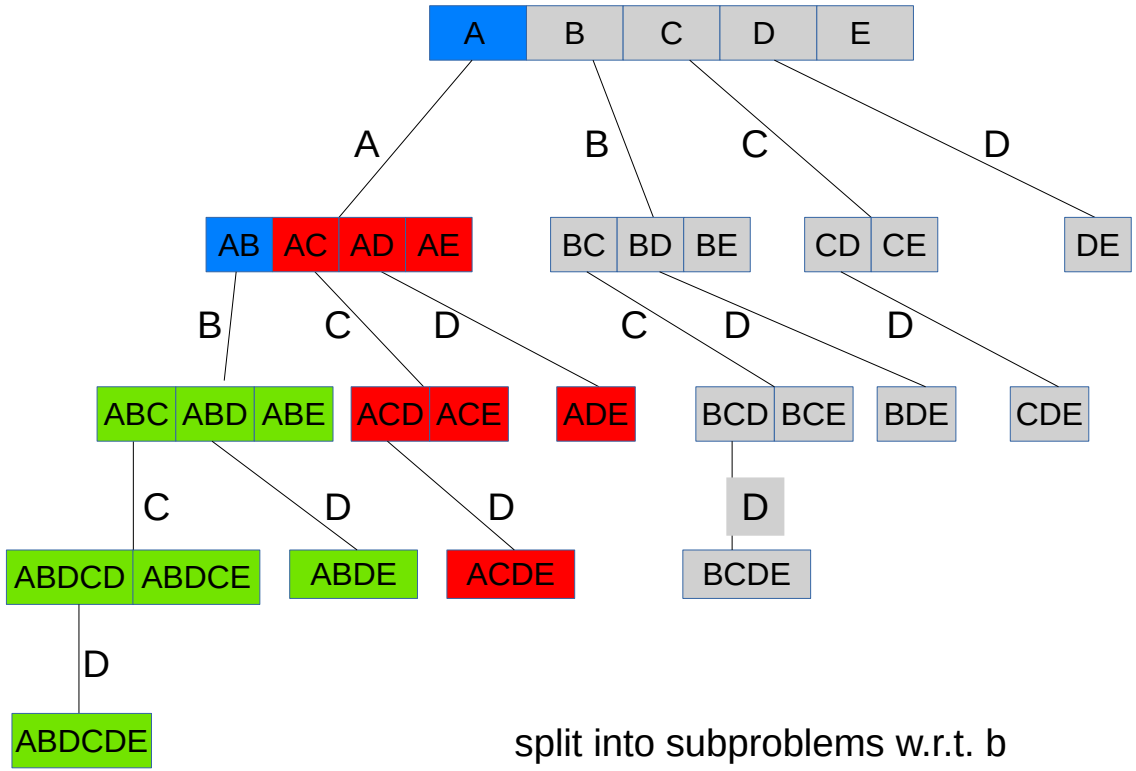


Depth-First Search and Conditional Databases



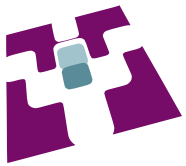
- blue: itemset containing only item *a*.
  green: itemsets containing item *a* (and at least one other item).
  red: itemsets not containing item *a* (but at least one other item).
- green: needs cond. database with transactions containing item *a*.
  red: needs cond. database with all transactions, but with item *a* removed.

Depth-First Search and Conditional Databases



split into subproblems w.r.t. b

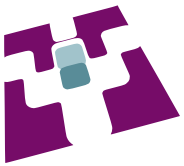
- blue: itemsets $\{a\}$ and $\{a, b\}$.
  green: itemsets containing items a and b (and at least one other item).
  red: itemsets containing item a (and at least one other item), but not item b.
- green: needs database with trans. containing both items a and b.
  red: needs database with trans. containing item a, but with item b removed.



The Eclat Algorithmt: Basic Ideas
[Zaki, Parthasarathy, Ogihara, and Li 1997]

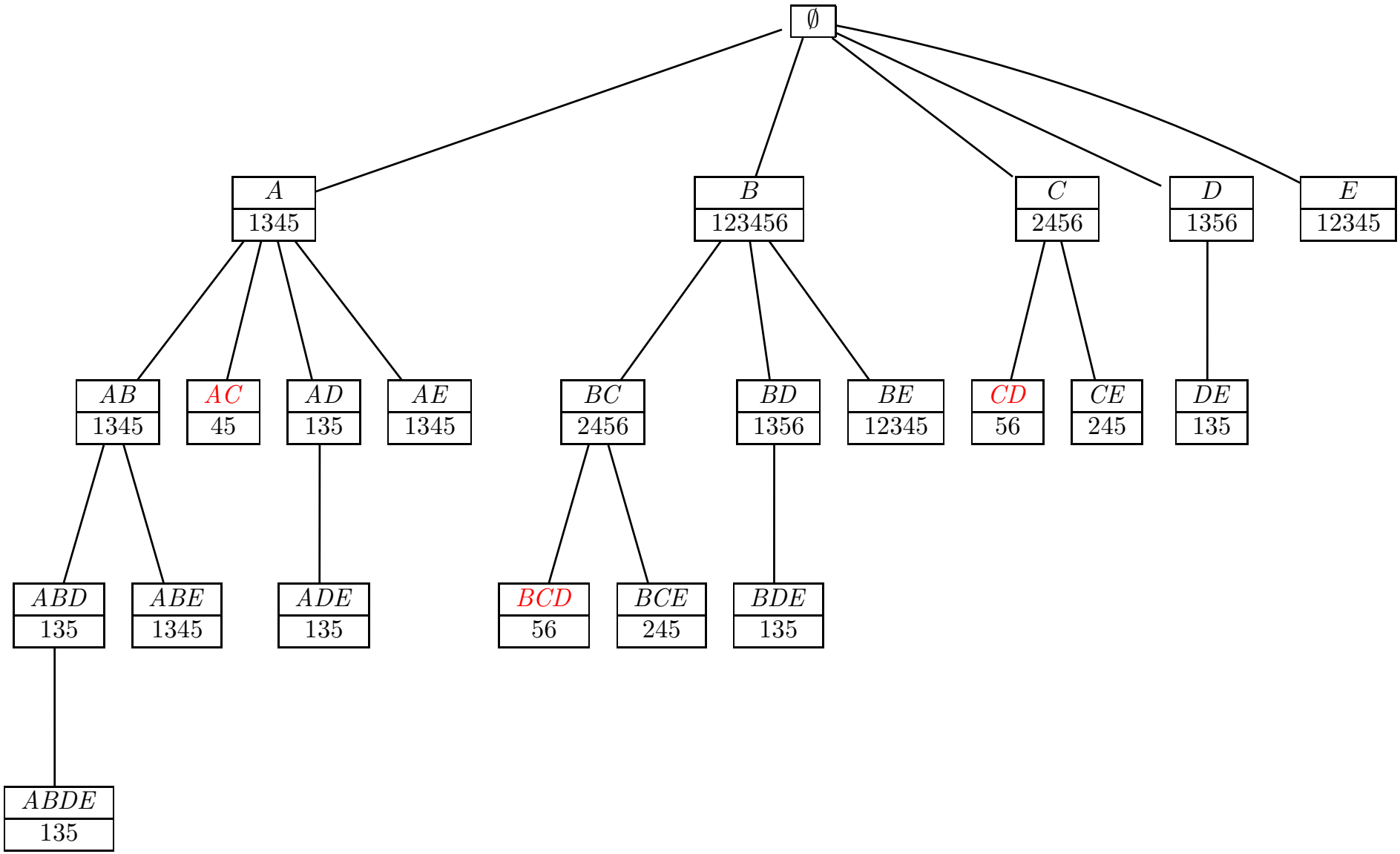
- The itemsets are checked in **lexicographic order** (**depth-first traversal** of the prefix tree).
- Eclat generates more candidate itemsets than Apriori, because it (usually) does not store the support of all visited itemsets. As a consequence it cannot fully exploit the Apriori property for pruning.
- Eclat uses a purely **vertical transaction representation**.
- No subset tests and no subset generation are needed to compute the support.

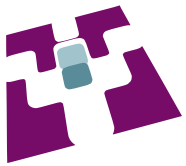
  The support of itemsets is rather determined by intersecting transaction lists.



Eclat Algorithm ($\theta = 3$)

Infrequent itemsets in red





Eclat Algorithm

// Initial Call: $\mathcal{F} \leftarrow \emptyset, P \leftarrow \{\langle i, \boldsymbol{t}(i)\rangle \mid i \in \mathcal{I}, |\boldsymbol{t}(i)| \geq \theta\}$

Eclat (P, θ, $\mathcal{F}$):

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

2 $\quad \mathcal{F} \leftarrow \mathcal{F} \cup \{(X_a, sup(X_a))\}$

3 $\quad P_a \leftarrow \emptyset$

4 $\quad$ **foreach** $\langle X_b, \boldsymbol{t}(X_b)\rangle \in P$, with $X_b > X_a$ **do**

5 $\quad\quad X_{ab} = X_a \cup X_b$

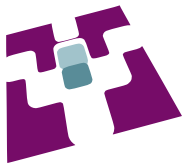
6 $\quad\quad \boldsymbol{t}(X_{ab}) = \boldsymbol{t}(X_a) \cap \boldsymbol{t}(X_b)$

7 $\quad\quad$ **if** $sup(X_{ab}) \geq \theta$ **then**

8 $\quad\quad\quad P_a \leftarrow P_a \cup \{\langle X_{ab}, \boldsymbol{t}(X_{ab})\rangle\}$

9 $\quad$ **if** $P_a \neq \emptyset$ **then** ECLAT $(P_a, \theta, \mathcal{F})$

10

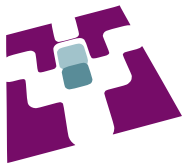


Pattern flooding

$\theta = 2$

$\mathcal{O}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
o_1	×	×	×	×	×										
o_2	×	×	×	×	×										
o_3	×	×	×	×	×										
o_4						×	×	×	×	×					
o_5						×	×	×	×	×					
o_6						×	×	×	×	×					
o_7											×	×	×	×	×
o_8											×	×	×	×	×

- How many frequent patterns?

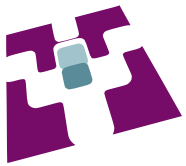


Pattern flooding

$\theta = 2$

$\mathcal{O}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
o_1	×	×	×	×	×										
o_2	×	×	×	×	×										
o_3	×	×	×	×	×										
o_4						×	×	×	×	×					
o_5						×	×	×	×	×					
o_6						×	×	×	×	×					
o_7											×	×	×	×	×
o_8											×	×	×	×	×

- How many frequent patterns? $1 + (2^5 - 1) \times 3 = 94$ patterns

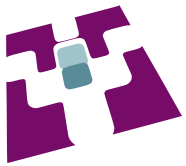


Pattern flooding

$\theta = 2$

$\mathcal{O}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
o_1	×	×	×	×	×										
o_2	×	×	×	×	×										
o_3	×	×	×	×	×										
o_4						×	×	×	×	×					
o_5						×	×	×	×	×					
o_6						×	×	×	×	×					
o_7											×	×	×	×	×
o_8											×	×	×	×	×

- How many frequent patterns? $1 + (2^5 - 1) \times 3 = 94$ patterns but actually 3 (potentially) interesting ones:
  $\{a_1, a_2, a_3, a_4, a_5\}, \{a_6, a_7, a_8, a_9, a_{10}\}, \{a_{11}, a_{12}, a_{13}, a_{14}, a_{15}\}$.



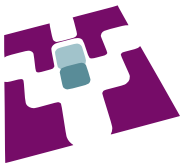
Pattern flooding

$\theta = 2$

$\mathcal{O}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
o_1	×	×	×	×	×										
o_2	×	×	×	×	×										
o_3	×	×	×	×	×										
o_4						×	×	×	×	×					
o_5						×	×	×	×	×					
o_6						×	×	×	×	×					
o_7											×	×	×	×	×
o_8											×	×	×	×	×

- How many frequent patterns? $1 + (2^5 - 1) \times 3 = 94$ patterns but actually 3 (potentially) interesting ones:
  $\{a_1, a_2, a_3, a_4, a_5\}, \{a_6, a_7, a_8, a_9, a_{10}\}, \{a_{11}, a_{12}, a_{13}, a_{14}, a_{15}\}$.

☞ the need to focus on a **condensed representation** of frequent patterns.



Maximal Frequent Itemsets

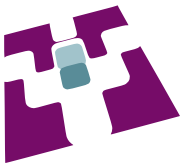
Given a binary database $\mathcal{D} \subseteq \mathcal{T} \times \mathcal{I}$, over the tids $\mathcal{T}$ and items $\mathcal{I}$, let $\mathcal{F}$ denote the set of all frequent itemsets, that is,

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

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.



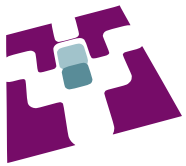
An Example Database

Transaction database

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

Frequent itemsets ($\theta = 3$)

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 Itemsets

- The set of all closed frequent itemsets is thus defined as

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

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

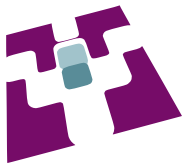
  it follows (can easily be proven by successively extending the itemset X)

$$\forall \theta : \forall X \in \mathcal{F} : \exists Y \in \mathcal{C} : \quad X \subseteq Y.$$

  That is: **Every frequent itemset has a closed superset.**

- Therefore:

$$\forall \theta : \quad \mathcal{F} = \bigcup_{X \in \mathcal{C}(\mathcal{D}, \theta)} 2^X$$



Closed Itemsets

- However, not only has every frequent itemset a closed superset, but it has a **closed superset with the same support**:

$$\forall\theta : \forall X \in \mathcal{F} : \exists Y \supseteq X : \quad Y \in \mathcal{C} \ \wedge \ \mathit{freq}(Y) = \mathit{freq}(X).$$

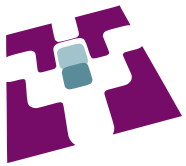
- The set of all closed itemsets preserves knowledge of all support values:

$$\forall\theta : \forall X \in \mathcal{F} : \quad \mathit{freq}(X) = \max_{Y \in \mathcal{C}, Y \supseteq X} \mathit{freq}(Y).$$

- Note that the weaker statement

$$\forall\theta : \forall X \in \mathcal{F} : \quad \mathit{freq}(X) \geq \max_{Y \in \mathcal{C}, Y \supseteq X} \mathit{freq}(Y)$$

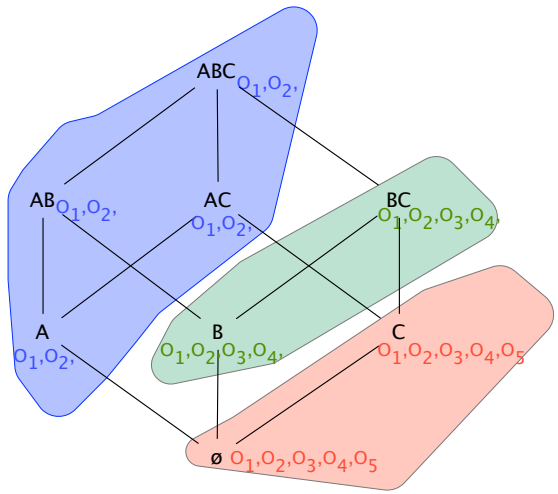
follows immediately from $\forall X : \forall Y \supseteq X : \mathit{freq}(X) \geq \mathit{freq}(Y)$, that is, an itemset cannot have a lower support than any of its supersets.

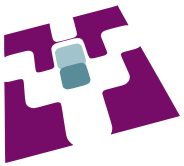


Closed and Free Patterns

Equivalence classes based on support.

$\mathcal{O}$	A	B	C
o_1	×	×	×
o_2	×	×	×
o_3		×	×
o_4		×	×
o_5			×

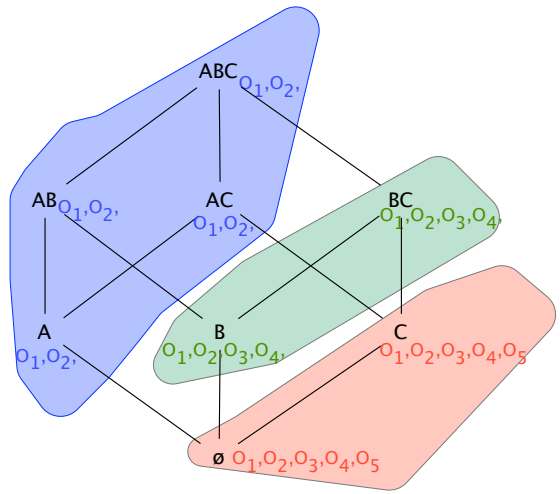




Closed and Free Patterns

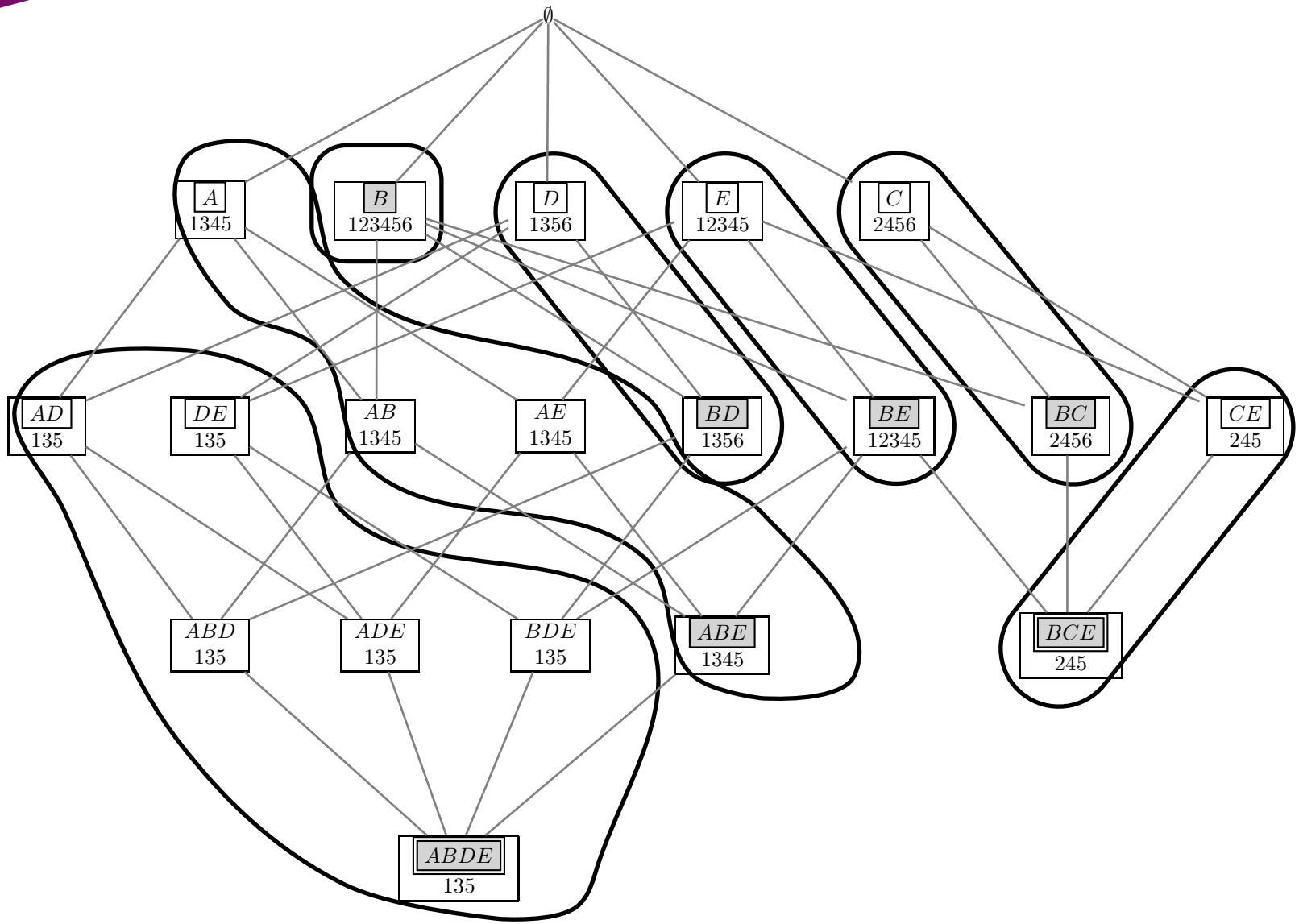
Equivalence classes based on support.

$\mathcal{O}$	A	B	C
o_1	×	×	×
o_2	×	×	×
o_3		×	×
o_4		×	×
o_5			×

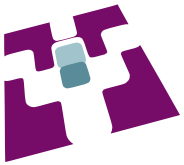


- **Closed** patterns are maximal element of each equivalence class (Bastide et al., SIGKDD Exp. 2000): ABC, BC, and C.
- **Generators** or **Free** patterns are minimal elements of each equivalent class (Boulicaut et al, DAMI 2003): $\{\}, A$ and B

Frequent Itemsets: Closed, Minimal Generators and Maximal



Itemsets boxed and shaded are closed, double boxed are maximal, and those boxed are minimal generators

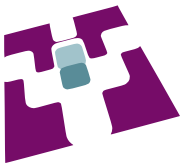


Mining Maximal Frequent Itemsets: GenMax Algorithm

Mining maximal itemsets requires additional steps beyond simply determining the frequent itemsets.

Assuming that the set of maximal frequent itemsets is initially empty ($\mathcal{M} = \emptyset$), 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

// Initial Call: $\mathcal{M} \leftarrow \emptyset$, $P \leftarrow \left\{\langle i, \boldsymbol{t}(i)\rangle \mid i \in \mathcal{I}, sup(i) \geq \theta\right\}$

GenMax (P, θ, $\mathcal{M}$):

1 $Y \leftarrow \bigcup X_i$

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

3 **return** // prune entire branch

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

5 $P_i \leftarrow \emptyset$

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

7 $X_{ij} \leftarrow X_i \cup X_j$

8 $\boldsymbol{t}(X_{ij}) = \boldsymbol{t}(X_i) \cap \boldsymbol{t}(X_j)$

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

10

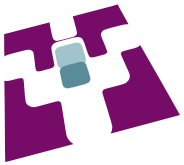
11 **if** $P_i \neq \emptyset$ **then** GenMax (P_i, θ, $\mathcal{M}$)

12

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

14 $\mathcal{M} = \mathcal{M} \cup X_i$ // add X_i to maximal set

15

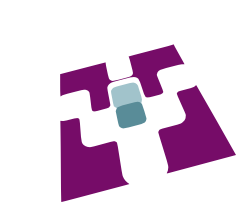


GenMax Algorithm: Maximal Itemsets

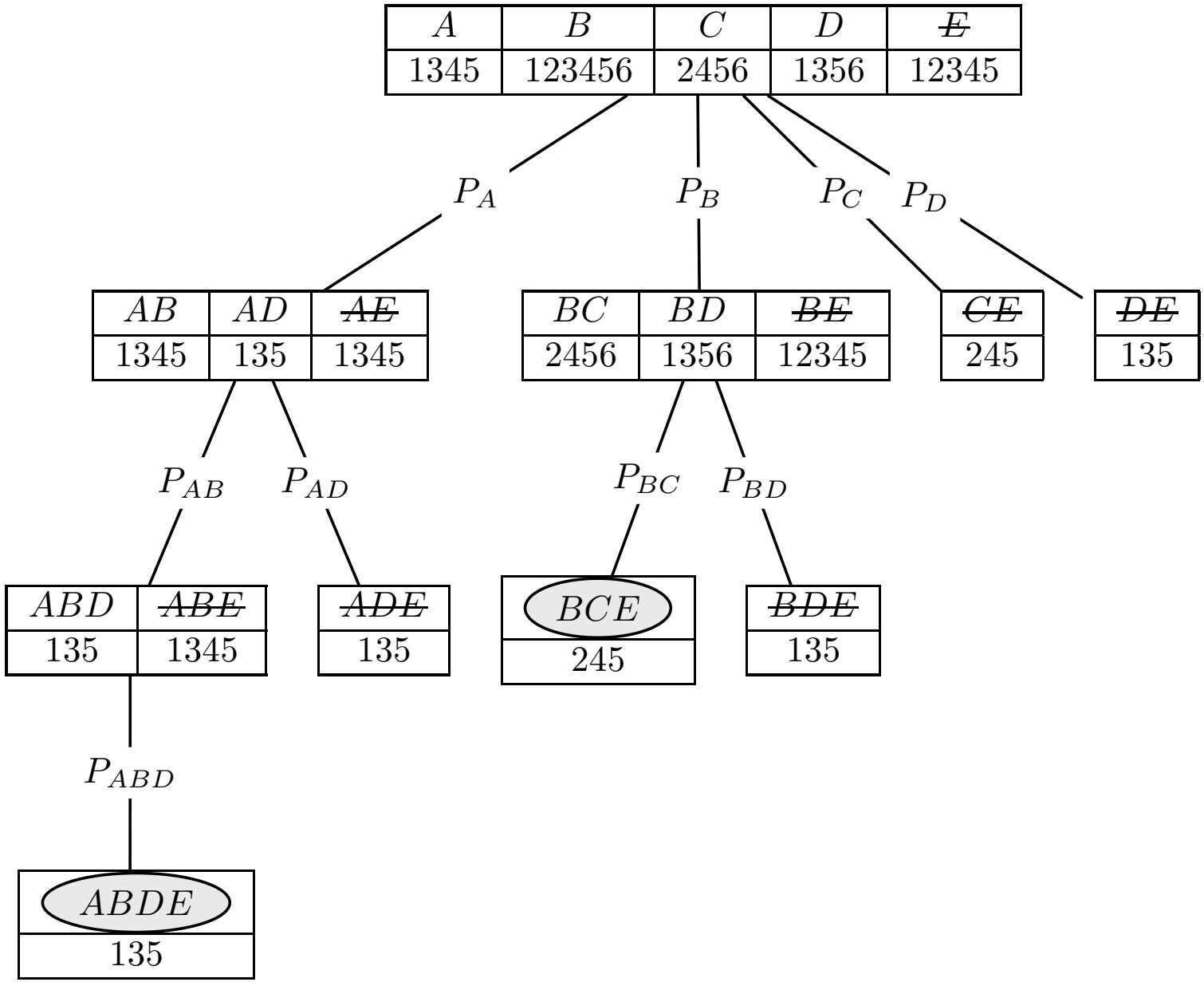
GenMax is based on dEclat. It takes as input the set of frequent items along with their tidsets, $\langle i, \boldsymbol{t}(i) \rangle$, and the initially empty set of maximal itemsets, $\mathcal{M}$.

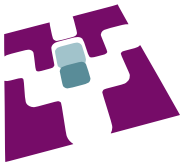
Given a set of itemset–tidset $\langle X, \boldsymbol{t}(X) \rangle$, GenMax method works as follows:

- If the union of all the itemsets, $Y = \bigcup X_i$, is already subsumed by some maximal pattern $Z \in \mathcal{M}$, then no maximal itemset can be generated from the current branch, and it is pruned. Otherwise, we 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.
- 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 $Z \in \mathcal{M}$.



Mining Maximal Frequent Itemsets





Association Rules

Often found patterns are expressed as **association rules**, for example:

> **If** *a customer buys* **bread** *and* **wine**,
> **then** *she/he will probably also buy* **cheese**.

An *association rule* is an expression $X \xrightarrow{s,c} Y$ where $X, Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$.
Let the itemset $X \cup Y$ be denoted as XY. The frequency of the rule is the number of transactions in which both X and Y co-occur as subsets:

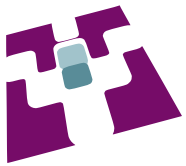
$$s = freq(X \longrightarrow Y) = freq(XY)$$

The relative support of the rule is defined as the fraction of transactions where X and Y co-occur, and it provides an estimate of the joint probability of X and Y:

$$rfreq(X \longrightarrow Y) = \frac{freq(XY)}{|\mathcal{D}|}$$

The confidence of a rule is the conditional probability that a transaction contains Y given that it contains X:

$$c = conf(X \longrightarrow Y) = P(Y|X) = \frac{P(X \wedge Y)}{P(X)} = \frac{sup(XY)}{sup(X)}$$



Example Dataset: Support and Confidence

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

Frequent itemsets: $\theta = 3$

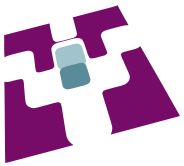
freq	*rfreq*	Itemsets
3	0.5	ABD, $ABDE$, AD, ADE BCE, BDE, CE, DE
4	0.67	A, C, D, AB, ABE, AE, BC, BD
5	0.83	E, BE
6	1.0	B

Rule confidence

Rule	*conf*
$A \longrightarrow E$	1.00
$E \longrightarrow A$	0.80
$B \longrightarrow E$	0.83
$E \longrightarrow B$	1.00
$E \longrightarrow BC$	0.60
$BC \longrightarrow E$	0.75

Care must be exercised in interpreting the goodness of a rule:

- $conf(E \longrightarrow BC) = 0.60$ (given E we have a probability of 60% of finding BC).
- but $P(BC) = 4/6 = 0.67$, which means that E has a deleterious effect on BC.



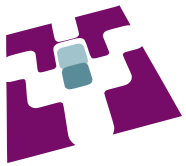
Rule Assessment Measures: Lift and Jaccard

Lift: Lift is defined as the ratio of the observed joint probability of X and Y to the expected joint probability if they were statistically independent, that is,

$$lift(X \longrightarrow Y) = \frac{P(XY)}{P(X) \cdot P(Y)} = \frac{rfreq(XY)}{rfreq(X) \cdot rfreq(Y)} = \frac{conf(X \longrightarrow Y)}{rfreq(Y)}$$

One common use of lift is to measure the surprise of a rule. A lift value close to 1 means that the support of a rule is expected considering the supports of its consequence.

Lift is **not downward closed**, that is, assuming that $X' \subset X$ and $Y' \subset Y$, it can happen that $lift(X' \longrightarrow Y') > lift(X \longrightarrow Y)$.



Lift, Support and Confidence

Rule	*lift*
$AE \longrightarrow BC$	0.75
$CE \longrightarrow AB$	1.00
$BE \longrightarrow AC$	1.20

Rule	*rfreq*	*conf*	*lift*
$E \longrightarrow AC$	0.33	0.40	1.20
$E \longrightarrow AB$	0.67	0.80	1.20
$B \longrightarrow E$	0.83	0.83	1.00

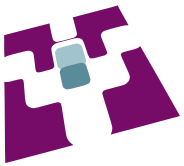
- $lift(AE \longrightarrow BC) = \frac{rfreq(ABCE)}{rfreq(AE) \cdot rfreq(BC)} = \frac{2/6}{4/6 \cdot 4/6} = 0.75$

  ➠ since the lift value is less than 1, the observed rule support is less than the expected support.

- $lift(BE \longrightarrow AC) = \frac{2/6}{2/6 \cdot 5/6} = 1.20$

  ➠ the observed rule occurs more than expected.

- Despite that $E \longrightarrow AC$ and $E \longrightarrow AB$ have lift greater than 1, $E \longrightarrow AC$ is weak rule (conf $= 0.4$), while $E \longrightarrow AB$ is not only stronger in terms of confidence, but it also have more support.



Jaccard

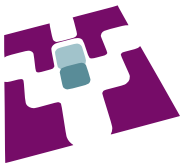
Jaccard: The Jaccard coefficient measures the similarity between two sets. When applied as a rule assessment measure it computes the similarity between the tidsets of X and Y:

$$jaccard(X \longrightarrow Y) = \frac{|\boldsymbol{t}(X) \cap \boldsymbol{t}(Y)|}{|\boldsymbol{t}(X) \cup \boldsymbol{t}(Y)|} = \frac{P(XY)}{P(X) + P(Y) - P(XY)}$$

Rule	*lift*
$AE \longrightarrow BC$	0.75
$CE \longrightarrow AB$	1.00
$BE \longrightarrow AC$	1.20

Rule	*rfreq*	*lift*	*jaccard*
$A \longrightarrow C$	0.33	0.75	0.33

$$jaccard(A \longrightarrow C) = \frac{freq(AC}{freq(A)+freq(C)-freq(AC)} = \frac{2}{4+4-2} = 0.33$$



Association Rules : Mining Task

Given:

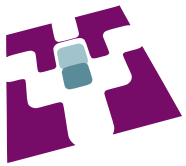
- a binary database $\mathcal{D} \subseteq \mathcal{T} \times \mathcal{I}$, over the tids $\mathcal{T}$ and items $\mathcal{I}$,
- a real number θ ➠ the **minimum frequency threshold**,
- a real number c_{min} ➠ the **minimum confidence**.

Desired: the set of all **association rules**, that is, the set

$$\mathcal{R} = \{R : X \longrightarrow Y \mid \mathit{freq}(R) \geq \theta \text{ and } \mathit{conf}(R) \geq c_{\text{min}}\}$$

General Procedure:

- Find the frequent item sets.
- Construct rules and filter them w.r.t. θ and c_{min}.



Properties of the Confidence

- From $\forall I : \forall J \subseteq I : \mathit{freq}(I) \leq \mathit{freq}(J)$ it obviously follows

$$\forall X, Y : \forall a \in X : \quad \frac{\mathit{freq}(XY)}{\mathit{freq}(X)} \geq \frac{\mathit{freq}(XY)}{\mathit{freq}(X - \{a\})}$$

and therefore

$$\forall X, Y : \forall a \in X : \quad \mathit{conf}(X \longrightarrow Y) \geq \mathit{conf}(X - \{a\} \rightarrow Y \cup \{a\})$$

That is: **Moving an item from the antecedent to the consequent cannot increase the confidence of a rule.**

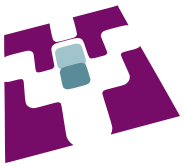
- As an immediate consequence we have $\forall X, Y : \forall a \in X :$

$$\mathit{conf}(X \longrightarrow Y) < c_{\text{min}} \longrightarrow \mathit{conf}(X - \{a\} \longrightarrow Y \cup \{a\}) < c_{\text{min}}$$

That is: **If a rule fails to meet the minimum confidence, no rules over the same item set and with a larger consequent need to be considered.**

GenRule ($\mathcal{F}$, θ, c_{min}, $\mathcal{R}$)**:**

1 $\mathcal{R} \leftarrow \emptyset$
2 **foreach** $f \in \mathcal{F}$ **do**
    // traverse the frequent item sets
3 $m \leftarrow 1$ // start with rule heads (consequents)
4 $H_m \leftarrow \bigcup_{i \in f} \{\{i\}\}$ // that contain only one item
5 **repeat**
6 **foreach** $h \in H_m$ **do**
7 **if** $(conf[(f - h) \longrightarrow h] \geq c_{min})$ **then**
8 $\mathcal{R} \leftarrow \mathcal{R} \cup \{[(f - h) \longrightarrow h]\}$ // add rule to the result
9 **else**
10 $H_m := H_m - \{h\}$ // otherwise discard the head
11 $H_{m+1} :=$ candidates(H_m) // create heads with one item more
12 $m \leftarrow m + 1$ // increment the head item counter
13 **until** $H_m = \emptyset$ **or** $m \geq |f|$
14 **return** $\mathcal{R}$

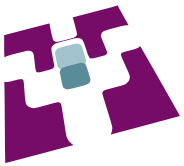


Frequent Item Sets: Example

t	$\boldsymbol{i}(t)$
1	ADE
2	BCD
3	ACE
4	$ACDE$
5	AE
6	ACD
7	BC
8	$ACDE$
9	BCE
10	ADE

0 items	1 item	2 items	3 items
$\emptyset$: 10	$\{a\}$: 7	$\{a, c\}$: 4	$\{a, c, d\}$: 3
	$\{b\}$: 3	$\{a, d\}$: 5	$\{a, c, e\}$: 3
	$\{c\}$: 7	$\{a, e\}$: 6	$\{a, d, e\}$: 4
	$\{d\}$: 6	$\{b, c\}$: 3	
	$\{e\}$: 7	$\{c, d\}$: 4	
		$\{c, e\}$: 4	
		$\{d, e\}$: 4	

- The minimum support is $\theta = 3$ or $c_{\text{min}} = 0.3 = 30\%$ in this example.
- There are $2^5 = 32$ possible item sets over $\mathcal{I} = \{a, b, c, d, e\}$.
- There are 16 frequent item sets (but only 10 transactions).



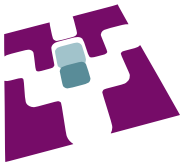
Generating Association Rules

Example: $f = \{A, C, E\}$, $X = \{C, E\}$, $Y = \{A\}$.

$$conf(CE \longrightarrow A) = \frac{freq(ACE)}{freq(CE)} = \frac{3}{4} = 75\%$$

Minimum confidence: 80%

association rule	support of all items	support of antecedent	confidence
$b \rightarrow c$	3 (30%)	3 (30%)	100%
$d \rightarrow a$	5 (50%)	6 (60%)	83.3%
$e \rightarrow a$	6 (60%)	7 (70%)	85.7%
$a \rightarrow e$	6 (60%)	7 (70%)	85.7%
$d, e \rightarrow a$	4 (40%)	4 (40%)	100%
$a, d \rightarrow e$	4 (40%)	5 (50%)	80%



Conclusions

Other well known modeling paradigm in A.I. : SAT, ILP, ASP ...

A growing interest for exploiting them in DM/ML:

- On wide range of tasks
- Reuse of solving technology: can outperform state-of-the-art

Nevertheless, scalability issue:

- Novel encodings/propagators
- Hybridization