Declarative Data Mining

Samir Loudni

TASC (LS2N – CNRS) – DAPI
IMT Atlantique

ACP'2020 GdR IA, RO, and ANITI Autumn School
26/11/2020

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 Item Set Mining : Motivation

Finding regularities from transaction databases

Example of **Market Basket Data**

- Finding regularities in the shopping behavior of customers of supermarkets, on-line shops, etc.
- More specifically:
  Find sets of products that are frequently bought together.
- Possible applications of found frequent item sets:
  - 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**.

Item Set: definition

Definition

Given a set of items (or attributes) $\mathcal{I}$, an itemset X is a subset of items, i.e., $X \subseteq \mathcal{I}$.

Input:

	i_1	i_2	$\ldots$	i_n
o_1	$d_{1,1}$	$d_{1,2}$	$\ldots$	$d_{1,n}$
o_2	$d_{2,1}$	$d_{2,2}$	$\ldots$	$d_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_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: $\mathcal{T} \subseteq \mathcal{O} \times \mathcal{I}$

	i_1	i_2	$\dots$	i_n
o_1	$d_{1,1}$	$d_{1,2}$	$\dots$	$d_{1,n}$
o_2	$d_{2,1}$	$d_{2,2}$	$\dots$	$d_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_m	$d_{m,1}$	$d_{m,2}$	$\dots$	$d_{m,n}$

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

Transactional representation: $\mathcal{T}$ is an array of subsets of $\mathcal{I}$

$$\begin{matrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{matrix}$$

where $t_i \subseteq \mathcal{I}$

Example

	i_1	i_2	i_3
o_1	×	×	×
o_2	×	×	
o_3		×	
o_4			×

t_1	i_1, i_2, i_3
t_2	i_1, i_2
t_3	i_2
t_4	i_3

Frequent Item Set mining

Problem Definition

Given the objects in $\mathcal{O}$ described with the Boolean attributes in $\mathcal{I}$, listing every item set having a frequency above a given threshold $\theta \in \mathbb{N}$.

Input:

	a_1	a_2	$\dots$	a_n
o_1	$d_{1,1}$	$d_{1,2}$	$\dots$	$d_{1,n}$
o_2	$d_{2,1}$	$d_{2,2}$	$\dots$	$d_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_m	$d_{m,1}$	$d_{m,2}$	$\dots$	$d_{m,n}$

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

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

Problem Definition

Given the objects in $\mathcal{O}$ described with the Boolean attributes in $\mathcal{I}$, listing every item set 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 Item Set mining: illustration

Specifying a minimal frequency threshold $\theta = 2$ objects (or, equivalently, a minimal relative frequency of 50%).

	a_1	a_2	a_3
o_1	×	×	×
o_2	×	×	
o_3		×	
o_4			×

Frequent Item Set mining: illustration

Specifying a minimal frequency threshold $\theta = 2$ objects (or, equivalently, a minimal relative frequency of 50%).

	a_1	a_2	a_3
o_1	$\times$	$\times$	$\times$
o_2	$\times$	$\times$	
o_3		$\times$	
o_4			$\times$

The frequent itemsets are: $\emptyset$ (4), $\{a_1\}$ (2), $\{a_2\}$ (3), $\{a_3\}$ (2) and $\{a_1, a_2\}$ (2).

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											×	×	×	×	×

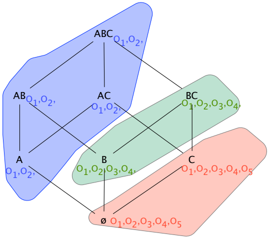
- How many frequent patterns? $1 + (2^5 - 1) \times 3 = 94$ patterns but actually 3 (potentially) interesting ones: $\{a_1, a_2, a_3, a_4, a_5\}, \{a_6, a_7, a_8, a_9, a_{10}\}, \{a_{11}, a_{12}, a_{13}, a_{14}, a_{15}\}$.

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

Closed and Free Patterns

Equivalence classes based on support.

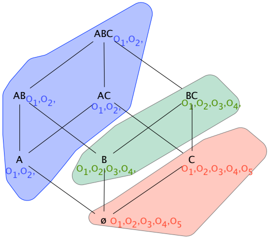
$\mathcal{O}$	A	B	C
o_1	×	×	×
o_2	×	×	×
o_3		×	×
o_4		×	×
o_5			×



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

- Consider the set of **closed (frequent) item sets**:

  $C(\mathcal{D}, \theta) = \{X \subseteq \mathcal{I} \mid freq(X, \mathcal{D}) \geq \theta \wedge \forall Y \supset X : freq(Y, \mathcal{D}) < freq(X, \mathcal{D})$

  That is: **An item set is closed if it is frequent, but none of its proper supersets has the same support.**

- With this definition it follows

$$\forall X \in F(\mathcal{D}, \theta) : \exists Y \in C(\mathcal{D}, \theta) : X \subseteq Y.$$

  That is: **Every frequent item set has a closed superset.**

Properties of the Support of Item Sets

- 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 item set'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 item set is extended, its support cannot increase.**

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

Properties of the Support of Item Sets

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

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

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

- Of course, the contraposition of this implication also holds:

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

  That is: **All subsets of a frequent item set are frequent.**

Why declarative approaches?

- Specific methods/algorithms for specific problems
  - Limited flexibility:
    - for each problem, do not write a solution from scratch
    - refining solution methods is hard, but typical in the KDD cycle
- Using constraint programming (CP) to specify data mining tasks as constraint satisfaction and optimization problems :
  - Reusing solving technology
  - Adding/removing (user) constraints
  - Exhaustive, optimal

Declarative approaches for Item set mining

Declarative Approaches

- Pattern mining : De Raedt et al., KDD'08, Lazaar et al., CP'16, Schaus et al., CP'17, Belaid et al., IJCAI'19, Jabbour et al.CIKM'13, Boudane et al., IJCAI'16, Y. Izza et al., IJCAI'20, A. Hien et al., ECML/PKDD'20
- Sequence mining : E. Coquery et al. ECAI'12, Negrevergne et al., CPAIOR'15, Kemmar et al. CP'15, Aoga et al. ECML/PKDD'16, A. Hosseininasab et al., AAAI'19
- Pattern set mining : Khiari et al., CP'10, Guns et al., TKDE'13, Ouali et al., PAKDD'17
- Skypatterns / multi-objective : Negrevergne et al., ICDM'16, Ugarte et al. ECAI'14 & AIJ'17
- Clustering : Mueller et al, DS'10, Babaki et al., CPAIOR'14, Dao et al. CP'15 & ECAI'16 & JCAI'18, Ouali et al. IJCAI'16, Chabert et al., CP'17 & JAIR'20, N. ARIBI et al. PAKDD'18
- Classification : H. Verhaeghe et al. IJCAI'20, A. Ignatiev et al., CP'20, M. Mulamba et al. CPAIOR'20

Constraint Programming

A generic framework for solving combinatorial problems

- A declarative description of the problem by a triplet (X, D, C) where
  - $X = \{x_1, \ldots, x_n\}$ is finite set of variables
  - $D = \{D_1, \ldots, D_n\}$ is finite set of domains (a.k.a possible values) of variables
  - $C = \{c_1, \ldots, c_e\}$ is a set of constraints restricting the values of variables x_i

- **Resolution** = Enumeration + Filtering

solution $\equiv$ assignments on X satisfying all constraints of C

Filering

- domains reduction : process of removing values from variables which cannot lead to any solution

- propagation : mechanism of calling the filtering algorithm associated with the constraints involving a variable x each time the domain of this variable is modified.

Filering

- domains reduction : process of removing values from variables which cannot lead to any solution
  - exemple: $x_1 > x_2$ et $D_1 = \{1, 2\}$, $D_2 = \{0, 1, 2, 3\}$
    ➠ $D_1 = \{1, 2\}$, $D_2 = \{0, 1, \not{2}, \not{3}\}$
- propagation : mechanism of calling the filtering algorithm associated with the constraints involving a variable x each time the domain of this variable is modified.

Resolution process

Filering

- domains reduction : process of removing values from variables which cannot lead to any solution
  - exemple: $x_1 > x_2$ et $D_1 = \{1, 2\}$, $D_2 = \{0, 1, 2, 3\}$
    ➠ $D_1 = \{1, 2\}$, $D_2 = \{0, 1, \not{2}, \not{3}\}$
- propagation : mechanism of calling the filtering algorithm associated with the constraints involving a variable x each time the domain of this variable is modified.
  - exemple: $x_1 > x_2$, $x_1 = x_3$ et $D_1 = \{1, 2\}$, $D_2 = \{0, 1, 2, 3\}$, $D_3 = \{1, 3\}$
    $(x_1 > x_2)$ ➠ $D_1 = \{1, 2\}$, $D_2 = \{0, 1, \not{2}, \not{3}\}$

Filering

- domains reduction : process of removing values from variables which cannot lead to any solution
  - exemple: $x_1 > x_2$ et $D_1 = \{1, 2\}$, $D_2 = \{0, 1, 2, 3\}$
    ➠ $D_1 = \{1, 2\}$, $D_2 = \{0, 1, \not{2}, \not{3}\}$
- propagation : mechanism of calling the filtering algorithm associated with the constraints involving a variable x each time the domain of this variable is modified.
  - exemple: $x_1 > x_2$, $x_1 = x_3$ et $D_1 = \{1, 2\}$, $D_2 = \{0, 1, 2, 3\}$, $D_3 = \{1, 3\}$
    $(x_1 > x_2)$ ➠ $D_1 = \{1, 2\}$, $D_2 = \{0, 1, \not{2}, \not{3}\}$
    $(x_1 = x_3)$ ➠ $D_1 = \{1, \not{2}\}$, $D_3 = \{1, \not{3}\}$

Resolution process

Filering

- domains reduction : process of removing values from variables which cannot lead to any solution
  - exemple: $x_1 > x_2$ et $D_1 = \{1, 2\}$, $D_2 = \{0, 1, 2, 3\}$
    ➠ $D_1 = \{1, 2\}$, $D_2 = \{0, 1, \cancel{2}, \cancel{3}\}$
- propagation : mechanism of calling the filtering algorithm associated with the constraints involving a variable x each time the domain of this variable is modified.
  - exemple: $x_1 > x_2$, $x_1 = x_3$ et $D_1 = \{1, 2\}$, $D_2 = \{0, 1, 2, 3\}$, $D_3 = \{1, 3\}$
    $(x_1 > x_2)$ ➠ $D_1 = \{1, 2\}$, $D_2 = \{0, 1, \cancel{2}, \cancel{3}\}$
    $(x_1 = x_3)$ ➠ $D_1 = \{1, \cancel{2}\}$, $D_3 = \{1, \cancel{3}\}$
    $(x_1 = 1) \wedge (x_1 > x_2)$ ➠ $D_2 = \{0, \cancel{1}\}$

Global Constraints

- Constraints defined by a relation on any number of variables
- Example: **AllDifferent**$(x_1, \ldots, x_n)$ specifies that all its variables must take different values
- Better filtering :
  - A level of consistency at least as high as one could maintain on elementary constraints
  - filtering performed using other tools
    - Algorithms on graphs / automatons,
    - network flow,
    - ...

Item set mining using CP
CP4IM (De Raedt et al., 2008)

- Let $\boldsymbol{d}$- be the 0/1 matrix where, for each transaction t and each item i, $(d_{t,i} = 1)$ iff $(i \in t)$.

- **Variables** :
  - Let X be the unknown pattern we are looking for. X is represented by n **Boolean variables** $\{X_i \mid i \in \mathcal{I}\}$ such that : $\forall i \in \mathcal{I}, (X_i = 1)$ iff $(i \in X)$
  - The support of pattern X is represented by m **Boolean variables** $\{T_t \mid t \in \mathcal{T}\}$ such that : $(T_t = 1)$ iff $(X \subseteq t)$

- **Constraints**:
  - coverage : $\forall t \in \mathcal{T}, (T_t = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} I_i \times (1 - d_{t,i}) = 0$
  - frequency : $\forall i \in \mathcal{I}, (X_i = 1) \Rightarrow \sum_{t \in \mathcal{T}} T_t \geq \theta$
  - redundant constraints : $\forall i \in \mathcal{I}, (X_i = 1) \Rightarrow \sum_{t \in \mathcal{T}} T_t \times d_{t,i} \geq \theta$

Example : CP model

Transaction database

	i_1	i_2	i_3
t_1	1	0	0
t_2	1	0	1
t_3	0	0	1
t_4	0	1	1

Coverage constraints

$$
\begin{array}{lcl}
T_1 = 1 & \Leftrightarrow & (X_2 + X_3 = 0) \\
T_2 = 1 & \Leftrightarrow & (X_2 = 0) \\
T_3 = 1 & \Leftrightarrow & (X_1 + X_2 = 0) \\
T_4 = 1 & \Leftrightarrow & (X_1 = 0)
\end{array}
$$

Frequency constraint

$$T_1 + T2 + T_3 + T_4 \quad \geq \quad \theta = 2$$

Redundant constraints

$$
\begin{array}{lcl}
X_1 = 1 & \Rightarrow & T_1 + T_2 \geq \theta = 2 \\
X_2 = 1 & \Rightarrow & T_4 \geq \theta \\
X_3 = 1 & \Rightarrow & T_2 + T_3 + T_4 \geq \theta = 2
\end{array}
$$

Example : Enumeration and propagation

a)

	0	i_2	i_3
t_1	1	0	0
t_2	1	0	1
t_3	0	0	1
$t_4 \leftarrow 1$	0	1	1

	0	0	i_3
t_1	1	0	0
$t_2 \leftarrow 1$	1	0	1
$t_3 \leftarrow 1$	0	0	1
$t_4 = 1$	0	1	1

	0	0	0
$t_1 \leftarrow 1$	1	0	0
$t_2 = 1$	1	0	1
$t_3 = 1$	0	0	1
$t_4 = 1$	0	1	1

	0	0	1
$t_1 \leftarrow 0$	1	0	0
$t_2 = 1$	1	0	1
$t_3 = 1$	0	0	1
$t_4 = 1$	0	1	1

b)

	1	0	0
$t_1 \leftarrow 1$	1	0	0
$2_1 \leftarrow 1$	1	0	1
$t_3 \leftarrow 0$	0	0	1
$t_4 \leftarrow 0$	0	1	1

Three frequent item sets mined:

- $\{i_1\}$ with cover $\{t_1, t_2\}$,
- $\{i_3\}$ with cover $\{t_2, t_3, t_4\}$
- $\emptyset$ with cover $\{t_1, t_2, t_3, t_4\}$.

- The closedness constraint ensures that a pattern has no superset with the same frequency.
  - coverage (required) : $\forall t \in \mathcal{T}, (T_t = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} I_i \times (1 - d_{t,i}) = 0$
  - closed : $\forall i \in \mathcal{I}, (X_i = 1) \Leftrightarrow \sum_{t \in \mathcal{T}} T_t \times (1 - d_{t,i}) = 0$

Reified model in a nutshell

Advantages:

- Intuitive CP encoding
- Generic: many constraints can be expressed
- Effective in case of tight constraints

Drawbacks:

- use an additional dimension of transaction variables
- huge number of constraints: (2n + m) reified constraints
- scalability issue: genericity/efficiency trade-off

➥ Scaling up CP solvers to large data ?

Global constraints for Closed Frequent item sets mining

A **global constraint** that encodes efficiently the Closed Frequent item sets mining problem (ClosedPatterns [Lazaar et al., 2016])

- Domain consistency with polynomial algorithm
- No reified constraints/extra variables

CoverSize global constraint (Schaus et al., CP'17) (not in this talk):

- New extra decision variable to manage the exact size of the cover of an itemset
- Offers more flexibility in modeling problems

$\textsc{ClosedPatterns}_{\mathcal{D},\theta}(X_1, \ldots, X_{|\mathcal{I}|})$ (1/2) (N. Lazaar et al., CP 2016)

- Use a vector X of Boolean variables $(X_1, \ldots, X_{|\mathcal{I}|})$ for representing item sets
- $\textsc{ClosedPatterns}_{\mathcal{D},\theta}(X)$ holds if and only if $freq(X) \geq \theta$ and X is closed
- Closure extension : A non-empty item set P is a closure extension of Q iff $cover(P \cup Q) = cover(Q)$ ➠ used for mining closed itemsets

  if P is a closure extension of Q, and none of the proper supersets of P is a closure extension of Q, then $P \cup Q$ forms a closed pattern.

Three filtering rules : Let X^+ be the set of present items

1. remove value 0 from $dom(X_i)$ if $\{i\}$ is a closure extension of X^+
2. remove value 1 from $dom(X_i)$ if the itemset $X^+ \cup \{i\}$ is infrequent w.r.t. θ
3. remove value 1 from $dom(X_i)$ if $cover(X^+ \cup \{i\}) \subseteq cover(X^+ \cup \{j\})$ where j is an absent item.

Time complexity: $O(n \times (n \times m))$

Running example : ClosedPatterns$_{\mathcal{D},3}(X_1, \ldots, X_6)$ (2/2)

Trans.	Items					
t_1	A	C		T	W	
t_2		C	D		W	
t_3	A	C		T	W	Z
t_4	A	C	D		W	Z
t_5	A	C	D	T	W	
t_6		C	D	T		

Trans.	A	C	D	T	W	Z
t_1	1	1	0	1	1	0
t_2	0	1	1	0	1	0
t_3	1	1	0	1	1	1
t_4	1	1	1	0	1	1
t_5	1	1	1	1	1	0
t_6	0	1	1	1	0	0

	0	0/1	0/1	0/1	1	0/1
Trans.	A	C	D	T	W	Z
t_1	1	1	0	1	1	0
t_2	0	1	1	0	1	0
t_3	1	1	0	1	1	1
t_4	1	1	1	0	1	1
t_5	1	1	1	1	1	0
t_6	0	1	1	1	0	0

Suppose that $X_1 = 0$ and $X_5 = 1$ ➟ $X = \{W\}$

Running example : ClosedPatterns$_{\mathcal{D},3}(X_1, \ldots, X_6)$ (2/2)

Trans.	Items					
t_1	A	C		T	W	
t_2		C	D		W	
t_3	A	C		T	W	Z
t_4	A	C	D		W	Z
t_5	A	C	D	T	W	
t_6		C	D	T		

Trans.	A	C	D	T	W	Z
t_1	1	1	0	1	1	0
t_2	0	1	1	0	1	0
t_3	1	1	0	1	1	1
t_4	1	1	1	0	1	1
t_5	1	1	1	1	1	0
t_6	0	1	1	1	0	0

	0	1	0/1	0/1	1	0/1
Trans.	A	C	D	T	W	Z
t_1	1	1	0	1	1	0
t_2	0	1	1	0	1	0
t_3	1	1	0	1	1	1
t_4	1	1	1	0	1	1
t_5	1	1	1	1	1	0
t_6	0	1	1	1	0	0

$\{C\}$ is a closure extension of $\{W\}$ ➠ Rule#1 applied

Running example : ClosedPatterns$_{\mathcal{D},3}(X_1, \ldots, X_6)$ (2/2)

Trans.	Items					
t_1	A	C		T	W	
t_2		C	D		W	
t_3	A	C		T	W	Z
t_4	A	C	D		W	Z
t_5	A	C	D	T	W	
t_6		C	D	T		

Trans.	A	C	D	T	W	Z
t_1	1	1	0	1	1	0
t_2	0	1	1	0	1	0
t_3	1	1	0	1	1	1
t_4	1	1	1	0	1	1
t_5	1	1	1	1	1	0
t_6	0	1	1	1	0	0

	0	1	0/1	0/1	1	0
Trans.	A	C	D	T	W	Z
t_1	1	1	0	1	1	0
t_2	0	1	1	0	1	0
t_3	1	1	0	1	1	1
t_4	1	1	1	0	1	1
t_5	1	1	1	1	1	0
t_6	0	1	1	1	0	0

$freq(WZ) = 2 < 3$ ➠ Rule#2 applied

Running example : ClosedPatterns$_{\mathcal{D},3}(X_1,\ldots,X_6)$ (2/2)

Trans.	Items					
t_1	A	C		T	W	
t_2		C	D		W	
t_3	A	C		T	W	Z
t_4	A	C	D		W	Z
t_5	A	C	D	T	W	
t_6		C	D	T		

Trans.	A	C	D	T	W	Z
t_1	1	1	0	1	1	0
t_2	0	1	1	0	1	0
t_3	1	1	0	1	1	1
t_4	1	1	1	0	1	1
t_5	1	1	1	1	1	0
t_6	0	1	1	1	0	0

	0	1	0/1	0	1	0
Trans.	A	C	D	T	W	Z
t_1	1	1	0	1	1	0
t_2	0	1	1	0	1	0
t_3	1	1	0	1	1	1
t_4	1	1	1	0	1	1
t_5	1	1	1	1	1	0
t_6	0	1	1	1	0	0

$cover(TW) \subset cover(AW)$ ➠ Rule#3 applied

Experiments (1/3)

Dataset	$\|\mathcal{T}\|$	$\|\mathcal{I}\|$	ρ	type of data
Chess	3 196	75	49%	game steps
Splice1	3 190	287	21%	genetic sequences
Mushroom	8 124	119	19%	species of mushrooms
Connect	67 557	129	33%	game steps
BMS-Web-View1	59 602	2.5	0.5%	web click stream
T10I4D100K	100 000	1 000	1%	synthetic dataset
T40I10D100K	100 000	1 000	4%	synthetic dataset
Pumsb	49 046	7 117	1%	census data
Retail	88 162	16 470	0.06%	retail market basket data

Comparison with:

- The most efficient CP method: CP4IM (reified)
- The most efficient ad hoc algorithm: LCM-v5.2ce.2cm

Implementation: `OR-Tools`, timeout = `3600 s.`
➥ `https://loudni.users.greyc.fr/CPMiner.html`

Experiments (2/3) : CLOSEDPATTERNS vs CP4IM

$\mathcal{D}$	θ (%)	$\#C$ ($\approx$)	#Nodes ClosedPatterns	#Nodes CP4IM	Time (s) ClosedPatterns	Time (s) CP4IM
chess	50	10^5	**738 901**	738 907	**3.21**	10.97
	40	10^6	**2 733 667**	2 733 735	**12.27**	40.85
	30	10^6	**10 679 631**	10 681 739	**45.92**	136.31
	20	10^7	**45 837 171**	45 901 933	**187.89**	467.52
	10	10^8	**247 960 091**	249 411 325	**969.40**	1 950.51
splice1	20	10^2	**487**	**487**	**0.59**	22.59
	10	10^3	**3 211**	**3 211**	**0.14**	25.54
	5	10^4	**63 935**	**63 935**	**3.23**	138.54
	1	10^7	**13 454 755**	13 467 247	**400.10**	1 652.41
connect	90	10^3	**6 973**	**6 973**	**0.92**	7.10
	80	10^4	**30 223**	**30 223**	**1.65**	16.57
	70	10^4	**71 761**	**71 761**	**4.09**	33.72
	60	10^5	**136 699**	**136 699**	**7.30**	45.73
	50	10^5	**260 223**	**260 223**	**14.53**	110.19
	40	10^5	**478 781**	**478 781**	**27.32**	153.39
	30	10^5	**920 823**	**920 823**	**49.97**	304.52
	20	10^6	**2 966 399**	**2 966 399**	**157.40**	712.68
	10	10^7	**16 075 555**	**16 075 555**	**760.71**	2 597.89
T40*	10	10^2	**177**	OOM	**1.13**	OOM
	5	10^2	**643**	OOM	**1.78**	OOM
	1	10^5	**130 477**	OOM	**25.78**	OOM
	0.5	10^6	**2 551 883**	OOM	**953.58**	OOM
retail	10	10	**19**	OOM	**2.55**	OOM
	1	10^2	**329**	OOM	**4.02**	OOM
	0.5	10^3	**1 233**	OOM	**12.73**	OOM
	0.1	10^4	**15 901**	OOM	**796.82**	OOM
	0.05	10^4	**40 229**	OOM	**2 645.06**	OOM

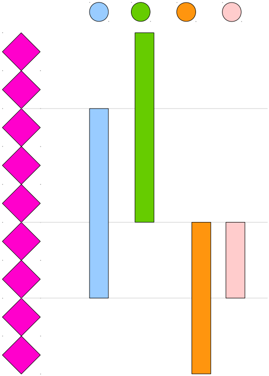
Experiments (3/3) : ClosedPatterns vs LCM

$\mathcal{D}$	θ	$\#C$	Time (s)		
	(%)	($\approx$)	ClosedPatterns	CP4IM	lcm
chess	50	10^5	3.21	10.97	**.32**
	40	10^6	12.27	40.85	**.44**
	30	10^6	45.92	136.31	**.07**
	20	10^7	187.89	467.52	**7.55**
	10	10^8	969.40	1 950.51	**41.55**
splice1	20	10^2	0.59	22.59	**.04**
	10	10^3	0.14	25.54	**.07**
	5	10^4	3.23	138.54	**.46**
	1	10^7	400.10	1 652.41	**3.59**
connect	90	10^3	0.92	7.10	**.22**
	80	10^4	1.65	16.57	**.31**
	70	10^4	4.09	33.72	**.40**
	60	10^5	7.30	45.73	**.39**
	50	10^5	14.53	110.19	**.52**
	40	10^5	27.32	153.39	**.83**
	30	10^5	49.97	304.52	**.37**
	20	10^6	157.40	712.68	**.37**
	10	10^7	760.71	2 597.89	**7.70**
T40*	10	10^2	1.13	OOM	**.43**
	5	10^2	1.78	OOM	**.31**
	1	10^5	25.78	OOM	**1.32**
	0.5	10^6	953.58	OOM	**3.31**
retail	10	10	2.55	OOM	**.06**
	1	10^2	4.02	OOM	**.10**
	0.5	10^3	12.73	OOM	**.32**
	0.1	10^4	796.82	OOM	**.80**
	0.05	10^4	2 645.06	OOM	**.07**

Mining diverse patterns using CP

Redundancy

- Too many patterns, unmanageable and diversity not necessary assured
- Find a set of patterns that is:
  - small
  - non-redundant
- Several approaches for mining non-redundant patterns :
  - Mikis (Knobbe & Ho, 2006)
  - Piker (Bringmann & Zimmermann, 2009)

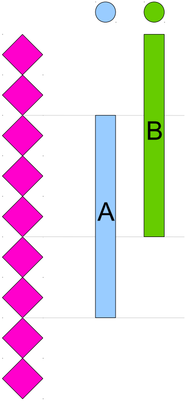


➠ Exploiting CP for mining diverse set of patterns

Measuring Redundancy

Exploiting similarity measure to compare pairs of patterns :

- $|A \cap B|$
- $|A \cap B| / |A|.|B|$ (Cosine similarity)
- $|A \cap B| / |A \cup B|$ (Jaccard index)
- $|A \cup B| - |A \cap B|$ (Hamming distance)



Definition (Diversity/Jaccard constraint)

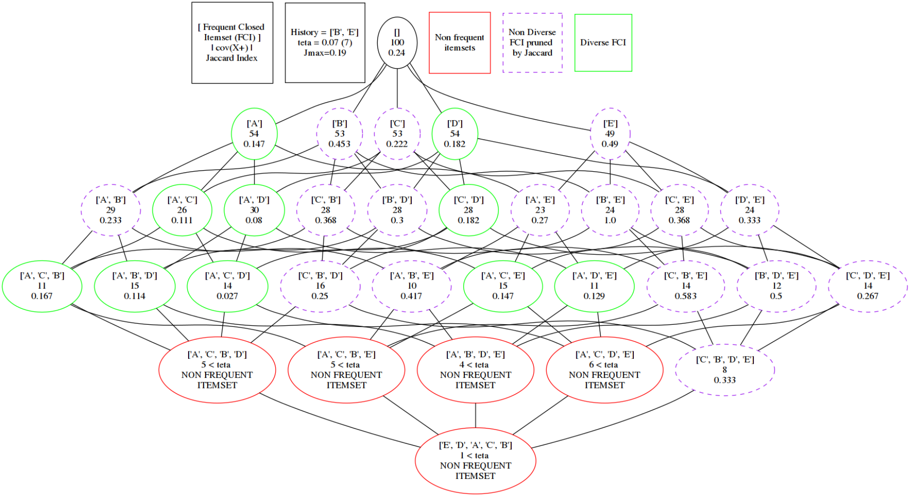
Let P and Q be two patterns. Given the Jaccard index Jac and a diversity threshold J_{max}, we say that P and Q are pairwise diverse iff $Jac(P, Q) \leq J_{max}$.

Idea: Push the Jaccard constraint during pattern discovery to prune non-diverse patterns.

Task : Given a history $\mathcal{H}$ of k pairwise diverse frequent closed patterns, the task is to mine new patterns P such that $\forall\, H \in \mathcal{H}$, $Jac(P, H) \leq J_{max}$.

Anti-monotonicity of the Jaccard constraint

The anti-monotonicity does not hold for the Jaccard constraint



- For $\mathcal{H} = \{BE\}$ and $J_{max} = 0.19$, $Jac(AE, \mathcal{H}) = 0.27 \geq J_{max}$ whereas $Jac(ACE, \mathcal{H}) = 0.147 \leq J_{max}$.

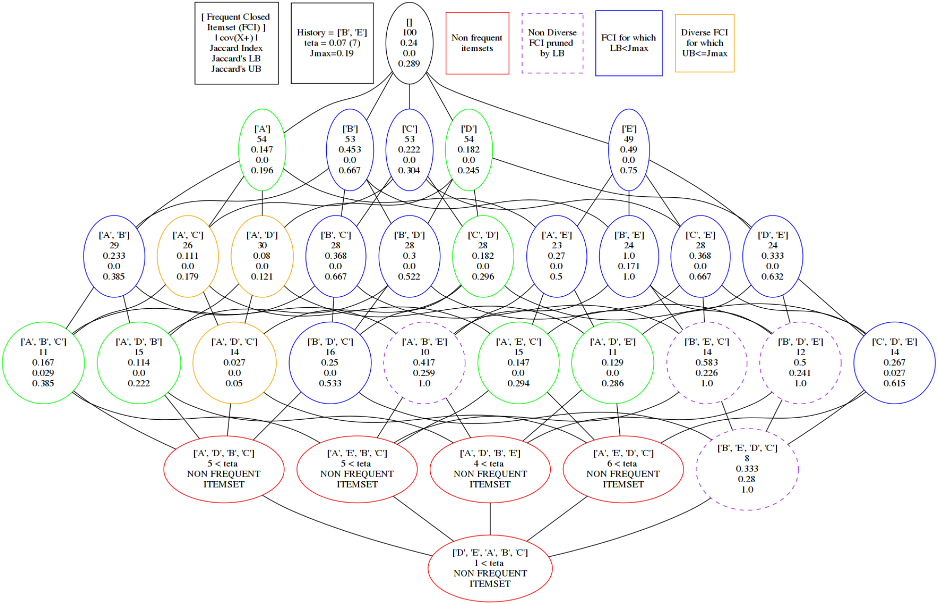
Relaxation of the Jaccard constraint

Anti-monotonic relaxations of the Jaccard constraint:

(i) **A lower bound relaxation** LB_J, which allows to prune non-diverse patterns during search;

(ii) **An upper bound relaxation** UB_J to find patterns ensuring diversity.

Relaxation of the Jaccard constraint : Example



Jaccard lower and upper bounds

- Monotonicity of LB_J: Let $H \in \mathcal{H}$ be an itemset. For any two patterns $P \subseteq Q$, the relationship $LB_J(P, H) \leq LB_J(Q, H)$ holds.

  ➥ If $LB_J(P, H) > J_{max} \Rightarrow Jac(P, H) > J_{max} \Rightarrow P$ **is not diverse**

- Anti-monotonicity of UB_J: Let $H \in \mathcal{H}$ be an itemset. For any two patterns $P \subseteq Q$, the relationship $UB_J(P, H) \geq UB_J(Q, H)$ holds.

  ➥ If $UB_J(P, H) \leq J_{max} \Rightarrow Jac(P, H) \leq J_{max} \Rightarrow P$ **is diverse**

- New mining task : Given a history $\mathcal{H}$ of k pairwise diverse frequent closed patterns, the new task is to mine candidate patterns P s.t. $\forall H \in \mathcal{H}$, $LB_J(P, H) \leq J_{max}$. When $UB_J(P, H) \leq J_{max}$, for all $H \in \mathcal{H}$, the Jaccard constraint is fully satisfied.

$\text{ClosedDiversity}_{\mathcal{D},\theta}(X, \mathcal{H}, J_{max})$ (A. Hien et al., ECML/PKDD 2020)

- Use a vector X of Boolean variables $(X_1, \ldots, X_{|\mathcal{I}|})$ for representing item sets
- $\text{ClosedDiversity}_{\mathcal{D},\theta}(X, \mathcal{H}, J_{max})$ holds if and only if :
  (1) $freq(X) \geq \theta$ and X is closed ➠ ClosedPatterns
  (2) X is diverse, $\forall\, H \in \mathcal{H}, LB_J(X, H) \leq J_{max}$.

Two filtering rules : Let X_{Div} be the set of items filtered by (Rule #1)

1. remove 1 from $dom(X_i)$ if $\exists\, H \in \mathcal{H}$ s.t. $LB_J(X^+ \cup \{i\}, H) > J_{max}$
2. remove 1 from $dom(X_i)$ if $\exists\, k \in X_{Div}$ s.t. $cover(X^+ \cup \{i\}) \subseteq cover(X^+ \cup \{k\})$
   ➥ $LB_J(X^+ \cup \{i\}, H) > LB_J(X^+ \cup \{k\}, H) > J_{max}$

Time complexity: $O(n \times (n \times m))$

Exploiting the upper bound to boost the search

New branching heuristic (WITNESS) : select the next free item i s.t.
$\forall H \in \mathcal{H}, UB_J(X^+ \cup \{i\}, H) \leq J_{max}$

Main idea : select free items favoring the satisfaction of the Jaccard constraint when extending the partial assignment X^+ of an itemset.

Exploring the witness subtree: as all supersets of $X^+ \cup \{i\}$ will satisfy the Jaccard constraint, only generate the first closed diverse pattern in the subtree, add it to the history $\mathcal{H}$ and continue the exploration of the remaining search space.

Baseline brunching heuristic (MINCOV) : select the next free item i having the minimum estimated support.

Experiments (1/3)

Dataset	$\|\mathcal{T}\|$	$\|\mathcal{I}\|$	ρ	type of data
Chess	3 196	75	49%	game steps
Splice1	3 190	287	21%	genetic sequences
Mushroom	8 124	119	19%	species of mushrooms
Connect	67 557	129	33%	game steps
BMS-Web-View1	59 602	2.5	0.5%	web click stream
T10I4D100K	100 000	1 000	1%	synthetic dataset
T40I10D100K	100 000	1 000	4%	synthetic dataset
Pumsb	49 046	7 117	1%	census data
Retail	88 162	16 470	0.06%	retail market basket data

Comparison with:

- ClosedPatterns (denoted ClosedP)
- Flexics (Dzyuba et al., DMKD 2017)

Implementation: `Choco` solver, timeout = `24 hours`

Comparing ClosedDiv with ClosedP (1/2)

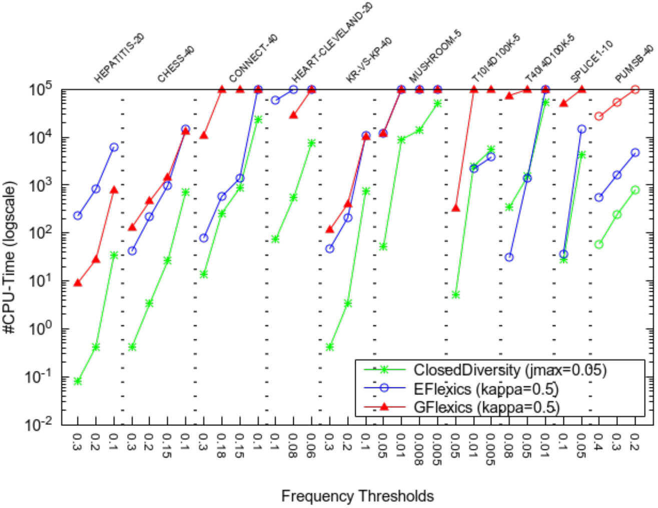
Dataset $\vert\mathcal{I}\vert \times \vert\mathcal{T}\vert$ $\rho(\%)$	$\theta(\%)$	#Patterns		Time (s)		#Nodes	
		(1)	(2)	(1)	(2)	(2)	(2)
Chess 75 × 3196 49.33%	30	5,316,468	**14**	815.15	**0.41**	10,632,935	**57**
	20	22,808,625	**65**	2838.30	**3.40**	45,617,249	**318**
	15	50,723,131	**238**	5666.03	**26.18**	101,446,261	**1,154**
	10	OOM	**1,622**	OOM	**728.13**	OOM	**7,774**
Kr-vs-kp 73 × 3196 49.32%	30	5,219,727	**14**	682.94	**0.41**	10,439,453	**57**
	20	21,676,719	**64**	2100.79	**3.41**	43,353,437	**307**
	10	OOM	**1,609**	OOM	**744.49**	OOM	**9,505**
Mushroom 112 × 8124 18.75%	5	8,977	**125**	**10.02**	52.21	17,953	**1,357**
	1	40,368	**9,935**	**34.76**	8976.82	80,735	**20,924**
	0.8	47,765	**12,743**	**36.52**	14136.48	95,529	**26,660**
	0.5	62,334	**23,931**	**50.05**	50646.09	124,667	**49,406**
Pumsb 2,113 × 49,046 3.50%	40	-	**4**	-	**58.78**	-	**15**
	30	-	**14**	-	**246.80**	-	**59**
	20	-	**39**	-	**797.87**	-	**206**
T40I10D100K 942 × 100000 4.20%	8	138	**125**	**75.91**	346.24	275	**249**
	5	317	**284**	**331.47**	1514.76	633	**567**
	1	65,237	**7,217**	5574.31	**53000.72**	130,473	**14,517**
Retail 16470 × 88,162 0.06%	5	17	**12**	**10.74**	31.13	33	**23**
	1	160	**105**	**297.21**	1599.69	319	**218**
	0.4	832	**515**	**6073.53**	31962.90	1,663	**1,071**

Table: (1): ClosedP (2): ClosedDiv ($J_{max} = 0.05$) with Mincov

Comparing ClosedDiv with ClosedP (2/2)

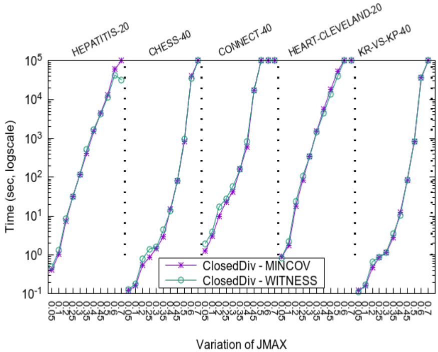
- ClosedDiv generates less patterns (in the thousands) in comparison to ClosedP (in millions).
- On dense data sets, ClosedDiv is up to an order of magnitude faster than ClosedP.
- On sparse data sets, ClosedDiv can take significantly more time to extract all diverse frequent closed patterns.

CPU Time Comparison of CLOSEDDIV with FLEXICS

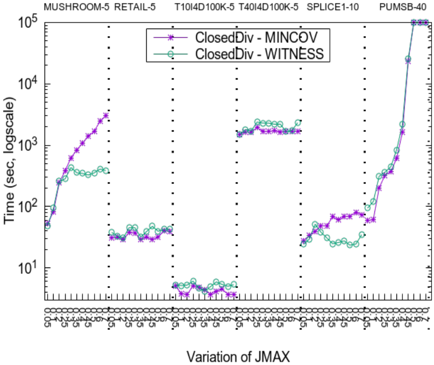


➟ CLOSEDDIV largely dominates FLEXICS, being more than an order of magnitude faster.

Witness vs Mincov: CPU-time



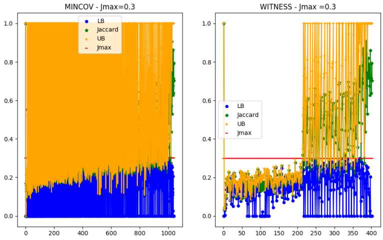
(a) Dense data sets.



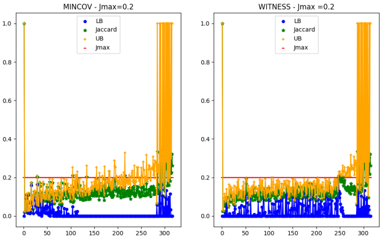
(b) Sparse data sets.

- The greater J_{max}, the longer the CPU time
- On dense data sets, both heuristics perform similarly
- On moderately dense data sets, Witness is very effective
- On sparse data sets, no heuristic clearly dominates the other

Qualitative analysis of the relaxation



(c) SPLICE1 ($\theta = 10\%, J_{max} = 0.3$)



(d) T40 ($\theta = 5\%, J_{max} = 0.2$)

➠ WITNESS allows to quickly discover a fewer set of patterns of better quality compared to MINCOV

Summary

➠ **A new generic solution for mining frequent diverse closed patterns**

- Exploiting relaxations in filtering and search procedure
- Other diversity measures (entropy)

➠ Leveraging Jaccard index in ClosedDiversity leads to pattern sets with more diversity among the patterns

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