Data Mining: Frequent Pattern Mining and Constraint-based Pattern Mining

M1 ENS

Let us consider the transaction database depicted in Tab. 1.

Id	Motif
1	$\{a, c, d\}$
2	$\{b, c, e\}$
3	$\{a, b, c, e\}$
4	$\{b, e\}$
5	$\{a, b, c, e\}$
6	$\{a, b, c, e\}$

Table 1: Transaction database

1. **Frequent Itemset Mining**
   (a) General questions independant of the minimum frequency threshold $minsup$:
      i. What is the maximal number of frequent itemsets that can be extracted from this dataset?
      ii. Draw the lattice of itemsets.
      iii. What is the maximal number of scans over the database with $APriori$ algorithm ? (breatdh-first enumeration)
   (b) Extract the frequent itemsets with $minsup = 2$ with Apriori Algorithm.
   (c) Compute the frequent itemsets ($minsup = 2$) by using a depth first strategy.
2. **Closed Frequent Itemset Mining and Formal Concept Analysis**
   A formal context is a triple $K = (G, M, I)$, where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a binary relation called incidence that expresses which objects have which attributes. The incidence relation can be regarded as a bipartite graph (or a partial order of height 2). Predicate gIm designates object g's having attribute m. For a subset $A \subseteq G$ of objects and a subset $B \subseteq M$ of attributes, one defines two derivation operators as follows:

   - $A' = \{m \in M | \forall g \in A, gIm\}$, and dually
   - $B' = \{g \in G | \forall m \in B, gIm\}$.

   Applying either derivation operator and then the other constitutes another operator, ″, with three properties (illustrated here for attributes):

   - idempotent: $A'''' = A''$,
   - monotonic: $A_1'' \subseteq A_2''$ whenever $A1 \subseteq A2$, and
   - extensive: $A \subseteq A''$.

   Any operator satisfying those three properties is called a **closure operator**, and any set A such that $A'' = A$ for a closure operator $''$ is called closed under $''$.

   With these derivation operators, it is possible to restate the definition of the term "formal concept" more rigorously: a pair (A,B) is a formal concept of a context (G, M, I) provided that:

   - $A \subseteq G$,

	m_1	m_2	m_3	m_4	m_5	m_6
g_1	×	×				×
g_2	×	×		×		×
g_3	×	×		×	×	×
g_4	×		×		×	
g_5	×				×	
g_6	×				×	×
g_7	×		×		×	×

Table 2: An example of formal context $\mathbb{K} = (G, M, I)$

- $B \subseteq M$,
- $A' = B$, and
- $B' = A$.

Equivalently and more intuitively, (A, B) is a formal concept precisely when: every object in A has every attribute in B, for every object in G that is not in A, there is some attribute in B that the object does not have, for every attribute in M that is not in B, there is some object in A that does not have that attribute. For a set of objects A, the set A' of their common attributes comprises the similarity characterizing the objects in A, while the closed set A" is the cluster of objects – within A or beyond – that have every attribute that is common to all the objects in A.

A formal context may be represented as a matrix K in which the rows correspond to the objects, the columns correspond to the attributes, and each entry $k_{i,j}$ is the boolean value of the expression "Object i has attribute j." In this matrix representation, each formal concept corresponds to a maximal submatrix (not necessarily contiguous) all of whose elements equal TRUE.

The Close by One algorithm[1] generates itemsets (concepts) in the lexicographical order of their extents assuming that there is a linear order on the set of objects. At each step of the algorithm there is a current object. The generation of a concept is considered canonical if its extent contains no object preceding the current object. Close by One uses the described canonicity test, a method for selecting subsets of a set of objects G and an intermediate structure that helps to compute closures more efficiently using the generated concepts. Its time complexity is $O(|G|^2|M||L|)$, and its polynomial delay is $O(|G|^3|M|)$ where |G| stands for the cardinality of the set of objects G, |M|, similarly, is the number of all attributes from M and |L| is the size of the concept lattice.

Example. Consider the set of objects $G = \{g_1, ..., g_7\}$ where each letter denotes an animal, respectively, "ostrich", "canary", "duck", "shark", "salmon", "frog", and "crocodile". Consider the set of attributes $M = \{m_1, .., m_6\}$ that are properties that animals may have or not, i.e. "borned from an egg", "has feather", "has tooth", "fly", "swim", "lives in air" . Table 2 gives an example of formal context (G, M, I) where I is defined by observing the given animals.

```
L = ∅
for each g ∈ G
    process({g}, g, (g'', g'))
L is the concept set.
```

Algorithm 1: Close By One.

(a) Which are the differences between formal concepts and closed itemsets?

(b) Apply Close by one algorithm to enumerate all frequent closed patterns (minsup=2). Push the minimum frequency constraint.

3. **Back to functional dependencies: Building Armstrong relations**

   Let R be a relation schema and F be a set of functional dependencies over R. An *Armstrong relation* for F is a relation r on R that fulfill only the functional dependencies from F^+. Let $R = ABCDE$ and $F = \{A \to BC, D \to E, C \to D\}$.

[1] Sergei O. Kuznetsov: A Fast Algorithm for Computing All Intersections of Objects in a Finite Semi-lattice. Automatic Documentation and Mathematical Linguistics, 1993.

if $\{h | h \in C \backslash A$ and $h < g\} = \emptyset$ **then**
  $L = L \cup \{(C, D)\}$
  for each $f \in \{h | h \in G \backslash C$ and $g < h\}$
    $Z = C \cup \{f\}$
    $Y = D \cap \{f'\}$
    $X = Y'$
    process$(Z, f, (X, Y))$
end if

Algorithm 2: process$(A, g, (C, D))$ with $C = A''$ and $D = A'$ and $<$ the lexical order on object names.

(a) Compute the set of closed of F defined as $Cl(F) = \{X^+ \mid X \subseteq R\}$.

(b) Compute the Armstrong relation r with algorithm 3.

(c) From the Armstrong relation, find some counter-examples for some functional dependencies not implied by F.

(d) From the Armstrong relation, exhibit some problems of redundancy.

Data: R a relation schema, F a set of functional dependencies over R.
Result: An Armstrong relation r w.r.t. F.
for $A \in R$ **do**
  $t[A] := 0$
end
$r := \{t\}$
$i := 1$
for $X \in Cl(F) \setminus R$ **do**
  for $A \in R$ **do**
    if $A \in X$ **then**
      $t[A] := 0$
    end
    else
      $t[A] := i$
    end
  end
  $r := r \cup \{t\}$
  $i := i + 1$
end
return r

Algorithm 3: Armstrong relation computation

4. **Constraint-based Pattern Mining**
Let us consider the following transaction database:

TID	Transactions
T_1	a,b,c,d,f
T_2	b,c,d,e,g
T_3	a,c,d,f
T_4	a,b,c,e,g
T_5	c,d,f,h

item	price
a	10
b	21
c	15
d	12
e	30
f	15
g	22
h	101

(a) We want to extract every pattern X which appears in at least two transactions ($support(X) \geq 2$) and whose items' price sum is lower than 40 ($sum(X) < 40$).
   i. What is the type of the constraint?
   ii. Enumerate all solutions.

(b) We now want to discover every pattern X which appears in at least two transactions ($support(X) \geq 2$) and whose average price is greater than 24 ($average(X) > 24$).
   i. What is the type of the constraint?
   ii. Convert the constraint into an anti-monotone constraint, then make the extraction.