Data Mining Lab 1

Esteban Marquer

November 2021

1 Intro

In this practical, we will see the basic pipeline to process a formal context with FCA, and code in Python two basic FCA algorithm.

We will be using 2 toy datasets from the theoretical class for this practical:

- a 4 by 4 boolean context: `shapes.csv`;
- a 9 by 26 multi-valued context: `banking.csv`.

As those files are CSV files, it is recommended to use the `pandas` library to load them, using `pandas.read_csv("shapes.csv", index_col=0)`.

2 Nominal, Ordinal, and Interordinal Scaling

The first thing to do to apply the basic FCA algorithms on data is to transform the multi-valued data into a boolean context. While `shapes.csv` is already a Boolean context, `banking.csv` is not.

We will see 3 scaling methods in this practical, for some attribute g:

- nominal scaling $K = (N, N, =)$:
  1. create a Boolean attribute g_i for each value i of the attribute g;
  2. for each object, set the value of the attribute g_i to $True$ if $g = i$;
- ordinal scaling $K = (N, N, \leq)$:
  1. create a Boolean attribute $g_{\leq i}$ for each value i of the attribute g;
  2. for each object, set the value of the attribute $g_{\leq i}$ to $True$ if $g \leq i$;
- interordinal scaling $K = (N, N, \leq \cup \geq)$:
  1. create 2 Boolean attributes $g_{\leq i}$ and $g_{\geq i}$ for each value i of the attribute g;
  2. for each object, set the value of the attribute $g_{\leq i}$ to $True$ if $g \leq i$ and similarly for $g_{\geq i}$ if $g \geq i$.

Note that nominal scaling works with categorical, ordinal or numerical data, while ordinal and interordinal scaling only make sense for ordinal or numerical data, as they rely on an order of the elements.

Implement for each of the three methods a function that will take a context (a `DataFrame` in our case) as input together with the name of an attribute (a character string). For ordinal and interordinal scaling, you will need an optional input with an order of the values (for example, a list of the values in increasing order). It should output the context with the attribute replaced by the corresponding scaling.

Apply your methods on each of the attributes of `banking.csv` to create a Boolean context (see slide 43). Which scaling do you use in each case? Why?

3 Implement the derivation and the closure operator

3.1 Derivation operator (optional)

Implement two functions corresponding to the derivation operator from respectively the set of objects and the set of attributes. The function should take a Boolean context and a set of objects as input and output a set of attributes, for the first function, and take a Boolean context and a set of attributes as input and output a set of objects for the second function.

Reminder of the derivation operator, with gIm if there is a cross[1] in the context for attribute g of object m:

$$\begin{aligned} A' &\stackrel{\text{def}}{=} \{m \in M \mid gIm \text{ for all } g \in A\}, \\ B' &\stackrel{\text{def}}{=} \{g \in G \mid gIm \text{ for all } m \in B\}. \end{aligned}$$

3.2 Closure operator

Implement the closure operator on the attributes of a context, by writing a function taking a context and a set of attributes as input and returning a set of attributes.

Reminder of the closure operator: $A'' = (A')'$.

Check that your code works correctly by computing $(A'')''$ for some set of attribute A, which should be equal to A'' (and to $((A'')'')''$ ...). Do so for `shapes.csv` and `banking.csv`.

4 Implement *AllClosure*

Using what you implemented before, we will implement the *AllClosure* algorithm seen in class.

[1] In a context with Boolean values, a cross corresponds to $True$ and no cross to $False$.

4.1 Implement *NextClosure* operator

Implement *NextClosure* (see slide 23).

4.2 Implement *AllClosure* using *NextClosure* operator

Implement *AllClosure* (see slide 22). It will take as an input a Boolean formal context. It should output a list of concepts (extent-intent pairs),

Hint: you will need to store the concepts in a list instead of outputing them each time like in 2.1. of slide 22.

5 Visualize with LatViz `https://latviz.loria.fr/`

Use `df_to_latviz` or `np_to_latviz` from the provided script `latviz_convert.py` to transform your context to fit the format of LatViz.

You should obtain the same results as with your algorithm.

6 Modify *AllClosure* to extract implications

You will now create a copy of your *AllClosure* algorithm and adapt it following slide 37 to create *CanonicalB*.

This *CanonicalB* algorithm computes the canonical basis of a context, which can be easier to interpret than concepts for decisions driven exploration.

7 Explore the dataset for the project

For the evaluation, you will use the French census data from 2016, for the Grand-Est region. It is available on Arche.

Together with the data you will find a PDF file in French describing the various attributes in the data and their values.

Your first job, should you choose to accept it, will be to select interesting attributes from all the available ones, and to select some of them that you will explore. For example, if I select the `NATNC` or `NATN49` attribute, I can wonder whether some other attributes allow to predict the birth nationality of a French resident.

You will then apply the methods mentioned in class and in the practical sessions to extract concepts, implication rules and other information and draw conclusion on them.

8 [Bonus] Implement Bordat

See `https://emarquer.github.io/docs/bordat.pdf` for a description of the Bordat algorithm in pseudocode.

Once you are done, compare the execution time and the results with the execution time of AllClosure.