ID2222 Data Mining Homework 2

GIOVANNI MANFREDI SEBASTIANO MENEGHIN
gioman | meneghin @kth.se

20th November 2023

1 Introduction

The homework consists of two tasks, one required and one extra:

Task1 : Implement the Apriori algorithm to find all the frequent item sets that have a support of at least s, i.e. they are large item sets. The output of this algorithm is L, the set of all large item sets.

Task2 : Implement an algorithm that finds all association rules in L that have a confidence of at least c.

To understand the assignment we remind the definition of **support** and **confidence**:

- **support** : characteristic of an **item set** equal to the number of transactions containing the item set. This is different from the definition seen in class, or on Agrawal paper that define this characteristic on a rule.
- **confidence** : characteristic of a **rule** $X \rightarrow Y$ equal to fraction of the number of transaction containing $X \cup Y$ over all transactions that contain X.

The environment chosen to implement these two algorithms is Python 3. The implementation consists of two Python files, namely *project_executor.py* and *classes.py*. The first file sets the command line parameters and runs the project, and the second file contains the two classes and the algorithm logic. These two classes are *Apriori* and *AssociationRules*. They respectively solve Task1 and Task2 of the project. They are explored in more detail in the following sections.

2 Apriori

This class solves Task1, taking as input a sale transaction data set "T10I4D100K.dat", (provided on Canvas, and available in the folder "data" of this project) and finding all item sets that have a support of at least s. The output of this task is L, a set of all large item sets of the initial data set.

To tackle this task, we divided the class into 5 methods:

1. **__init__**(self, data, s) : an initialisation method that takes as input. *data*, i.e. the file directory and the support s (a number of transactions inputted by the user).
2. **first_pass**(self, item) : a method that counts once, incrementing the counter in *C_k[item]*. This method is used for when the items of all item sets are passed the first time.
3. **apriori_gen**(self, L, k) : a method that takes as input a set L of candidates of dimension $k-1$ and k that sets the dimension of the candidates that we want. This method works with candidates of $k-1$ dimension to generate candidates of dimension k. These are returned as *C_k*.

4. **get_subsets**(self, Ck, t, k) : a method responsible for generating all item sets of dimension *k* from a set of candidate item sets *Ck*. To do this takes as input *Ck*, which represents the set of candidate itemsets, *t*, which represents a transaction or basket, and *k*, which represents the size of the subsets to be generated.

5. **algorithm**(self, verbose) : the actual implementation of the **Apriori algorithm** that calles previously described methods when needed. The algorithm proceeds by steps:

   (a) Goes through all baskets, counting all items and saving the count in *C_k* (uses **first_pass**).

   (b) Saves all item sets of dimension 1, i.e. that contain one item, that have support of at least *s*.

   (c) For the length of collection *L*, the algorithm generates all possible combinations starting from *L_1* that are subsets with support of at least *s* (uses **apriori_gen** and **get_subsets**).

   (d) Returns *L*.

3 AssociationRules

This class solves Task2, taking as input *L*, output of the Apriori algorithm (see above) and the confidence level *c*. The class method called **find** finds all association rules in *L* with a confidence of at least *c*. The output of the function find is a list of all association rules with a confidence of at least *c*.

To tackle this task, we implemented one method:

1. **find**(self, L, c, verbose, option) : the method takes several parameters (verbose is ignored, because it is explained later):

   - L: A list of dictionaries representing the frequent itemsets.
   - c: The minimum confidence threshold for generating association rules.

   The function iterates on *L* generating each possible rule on item sets of dimension of at least 2. Then the program checks if the rule has a confidence of at least *c* using the definition.

4 Results

The project runs on the file *project_executor.py* that takes in input a number of parameters from the command line:

- **-dataset-file** : sets the file directory. Defaults at the local directory "homework2/data/T10I4D100K.dat".
- **-s** : sets the minimum support. Note that it is a number of transactions not a percentage. Defaults at 1000 .
- **-c** : sets the minimum confidence. Note that it is a percentage. Defaults at 0.5 .
- **-verbose** : Boolean value that allows to print the results of the experiment. Defaults at TRUE.
- **-h** : help, i.e. describes parameters.

The project was tested on a conda environment run on a MacBook Pro with macOS Sonoma 14.0, 16 GB of RAM and an Apple Silicon M1 PRO, 2021.

In Figure 1 we can find the screenshot of the results of the project with the default values.

```
❯ /Users/giovannimanfredi/opt/miniconda3/envs/dm/bin/python /Users/giovannimanfredi/Developer/Projects/KTH/DM/Python/dm-2023-manfredi-meneghin/homework2/project_executor.py
Namespace(dataset_file='homework2/data/T10I4D100K.dat', s=1000, c=0.5, verbose=True)
1 - itemset time 0.21302199363708496 size of L_1  is  375
2 - itemset time 1.0185298919677734 size of L_2  is  9
3 - itemset time 0.7669599056243896 size of L_3  is  1
time for sub problem 1 3.9019477367401123
(704,) -> 39
(227,) -> 390
(704,) -> 825
(39, 704) -> 825
(39, 825) -> 704
(704, 825) -> 39
time for sub problem 2 4.696846008300781e-05
```

Figure 1: Timings and results for project_executor.py

In Figure 2 you can find the output generated by calling *-h* on the program.

```
❯ /Users/giovannimanfredi/opt/miniconda3/envs/dm/bin/python /Users/giovannimanfredi/Developer/Projects/KTH/DM/Python/dm-2023-manfredi-meneghin/homework2/project_executor.py -h
usage: project_executor.py [-h] [-dataset-file DATASET_FILE] [-s S] [-c C] [-verbose VERBOSE]

Find frequent itemsets and association rules for a given support/confidence

options:
  -h, --help            show this help message and exit
  -dataset-file DATASET_FILE
                        name of a transactions dataset with baskets and items
  -s S                  minimum support a itemset must have to be considered frequent
  -c C                  minimum confidence a rule must have to be generated
  -verbose VERBOSE      decides if the results are printed
```

Figure 2: Terminal commands

5 Running the project & terminal commands

To run the project no library outside the standard Python library is required. The only need is having a Python3 environment, that needs to be called when running the project. In our case a conda environment with a Python version 3.12 was tested.

The project can easily be run through the command line using the command *python3 project_executor.py* and specifying afterwards eventual non-default parameters.

Note that this running environment assumes the reader has downloaded a zipped version of the project and then extracted it. The default values in that case should allow to run the project smoothly.