Association mining with PYTHON

Objective: The objective of this exercise is to understand association mining, how frequent itemsets can be extracted by the Apriori algorithm and be able to calculate and interpret association rules in terms of support and confidence.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 12 folder available from Campusnet.

Piazza discussion forum: You can get help by asking questions on Piazza: https://piazza.com/dtu.dk/fall2020/02450

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $y_n \in \{0, 1, \dots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

12.1 Association Analysis

In this last exercise we will focus on association analysis. Association analysis is widely used in data mining in order to identify important co-occurrence relationships. We will use the following definition of association rule discovery:

> **Association Rule Discovery.** Given a set of transactions T, find all the rules having support $\geq$ *minsup* and confidence $\geq$ *minconf*, where *minsup* and *minconf* are the corresponding support and confidence thresholds.

We have summarized the most important terms in table 1. We will use the Apriori algorithm to find all itemsets with support greater than $\geq$ supp. The Apriori algorithm is based on the following principle:

> **Apriori principle.** If an itemset is frequent, then all of its subsets must also be frequent.

As a result of the Apriori principle we can start looking at frequent 1-itemsets. The frequent 2-itemsets can then only contain the items in the extracted 1-itemsets and so on and so forth. This greatly reduces the number of itemsets to check to find all frequent itemsets.

12.1.1 In table 2 some of the courses that 6 students completed during their studies are given. Find all itemsets with supp $\geq 80\%$.

12.1.2 What is the confidence of the rule 02457 $\leftarrow$ 02450?

We will use the Apriori algorithm to automatically mine for associations[1]. To use the Apriori algorithm, simply install the package `apyori` using `conda install apyori` (or `pip install apyori` if you are not using Anaconda).

12.1.3 Inspect the file `Data/courses.txt` and the script `ex12_1_3.py`. The script loads the course data file from table 2. Make sure you understand how the data in table 2 is stored in the file and how the script transforms it

12.1.4 Inspect and run the script `ex12_1_4.py`. The script transforms the binary matrix, plus label information, into a set of transactions. Make sure you understand how this transformation performs and relate it to the notation of the lecture notes. Finally note how the Apriori algorithm is invoked to find association rules with supp $\geq 80\%$ and conf $\geq 100\%$ and print them. Inspect the print command to see how you can access each association rule programmatically. What are the generated association rules?

[1] A high-performing version of the Apriori algorithm is also available from: `http://www.borgelt.net/apriori.html`.

Term	Meaning
$I = \{i_1, i_2, \ldots, i_d\}$	The set of all items
$T = \{t_1, t_2, \ldots, t_N\}$	The set of all transactions
Transaction, t_i	A subset of items: What was bought by a customer
Transaction width	Number of items in transaction
Itemset	A set of items from the set I of all items
k-itemset	An itemset having k items
Support count, $\sigma(X)$	Number of transactions that contain a particular itemset $\sigma(X) = \lvert\{t\}\rvert$
Association rule	Implication expression of the form $X \leftarrow Y$ where $X \bigcap Y = \varnothing$
Support, $s(Y \leftarrow X)$	Strength of association rule, $s(Y \leftarrow X) = \frac{\sigma(X \bigcup Y)}{N} = P(X, Y)$
Confidence, $c(Y \leftarrow X)$	Frequency items in Y appear in transactions containing X, $c(Y \leftarrow X) = \frac{\sigma(X \bigcup Y)}{\sigma(X)} = \frac{P(X,Y)}{P(X)} = P(Y\|X)$
Support-based pruning	Pruning strategy based on the Apriori principle (formed by the anti-monotone property)
Anti-monotone property	The support for an itemset never exceeds the support for its subsets (Apriori principle)
F_k	The set of frequent k-itemsets

Table 1: Association mining nomenclature.

	02322	02450	02451	02453	02454	02457	02459	02582
student 1	0	1	0	0	1	1	1	1
student 2	1	1	1	0	0	1	1	1
student 3	0	1	0	1	0	1	0	1
student 4	0	0	1	0	0	1	1	0
student 5	0	1	0	0	0	1	1	0
student 6	0	1	1	0	0	1	1	1

Table 2: Students that upon completing their engineering degree had taken various of the courses 02322, 02450, 02451, 02453, 02454, 02457, 02459 and 02582.

We will in this last part of the exercise mine for associations in the wine data [1](`http://archive.ics.uci.edu/ml/datasets/Wine+Quality`) considered in the previous exercises. However, as this data is not binary we will need to convert it to a format suitable for association mining. We will thus binarize the data by dividing each attribute into given percentiles.

12.1.5 Inspect and run the script `ex12_1_5.py`. The script load the `Data/wine2.mat` data into Python (for how to load .mat files into Python see also exercise 4.2.1) and divide each of the attributes in the data into percentiles using the function `binarize2`.

The scripts convert the continuous attributes into a one-out-of-K coding based on percentiles. Note how the function also transforms the attribute names. Why do you think we don't just include the 50-100 percentiles of a variable? What are the benefits of including variables corresponding to the 0-50 percentile?

12.1.6 Inspect and run the script `ex12_1_6.py` to find association rules in the Wine dataset with supp $\geq 30\%$ and conf $\geq 60\%$. Do these association rules make sense?

12.1.7 Often we are interested in rules with high confidence. Is it possible for itemsets to have very low support but still have a very high confidence?

12.1.8 (optional) Try find associations also in terms of the type of wine by adding two additional columns to the binary data corresponding to `1-y` and `y` (Note this is easiest done by adding new columns to the X-matrix and changing the `attributeNames` variable.)

References

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *In Decision Support Systems, Elsevier*, 47(4):547–553, 2009.