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"* C19 as well as the files in the exercise 12 folder available from Campusnet.

Preparation: Exercises 1-11

Part 1: Group discussion (max 15 min)

For the group discussion, each group should have selected a *discussion leader* at the previous exercise session. The purpose of the discussion leader is to ensure all team members understands the answers to the following two questions:

Multiple-Choice question: Solve and discuss **problem 19.1** from chapter 19 of the lecture notes. Ensure all group members understand the reason why one of the options is true and why the other options can be ruled out. (After today's exercises make sure to complete the remaining multiple-choice problems listed as part of the preparation for week 12 on the course homepage).

Discussion question: Discuss the following question in the group

- Consider the market-basket problem with 10 customers and 10 items from the slides. There are 1024 possible itemsets, why could we so quickly rule out most of them? Re-do the solution in the group. Can you find an association rule of the form $\{\text{beer}\} \rightarrow ?$ with confidence greater than .5?

Part 2: Programming exercises

Piazza discussion forum: You can get help by asking questions on Piazza: https://piazza.com/dtu.dk/fall2017/02450

Software installation: Extract the Python toolbox from Campusnet. 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, $\mathtt{y}_n \in \{0, 1, \ldots, C-1\}$, 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 part of the 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.

There is quite a bit of nomenclature in association mining, we have summarized the most important terms in table 1. In order to discover association rules the Apriori algorithm is commonly invoked. Here all itemsets with support $\geq$ *minsup* are identified based on the Apriori 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 search space to evaluate in order to identify all frequent itemsets.

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, $c(Y \leftarrow X) = \frac{\sigma(X \bigcup Y)}{\sigma(X)} = \frac{P(X,Y)}{P(X)} = P(Y\lvert X)$
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.

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

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

We will use the Apriori algorithm to automatically mine for associations. The Apriori algorithm we use is provided by `http://www.borgelt.net/apriori.html`, for details of the algorithm see also `http://www.borgelt.net/doc/apriori/apriori.`

	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.

`html`. You can use the script `run_apriori.py` from the toolbox, that enable to call the algorithm from Python. Look at the script `run_apriori.py` to see how the external algorithm can be called from the Python environment.

12.1.3 Inspect the file `Data/courses.txt` and make sure you understand how the data in table 2 is stored in the text file.

12.1.4 We will analyze the data in table 2 automatically using the script `runApriori.py`. Analyze the data with $minsupport \geq 80\%$ and $minconfidence \geq 100\%$, i.e. set the values of `minSup` and `minConf` appropriately and execute the script. What are the generated association rules?
(Notice if you are running Python on a linux computer you need to give permission to execute the file containing the apriori algorithm by typing `chmod +x apriori`). (Notice, if you are using Mac (OSX), then you may need to use `aprioriMAC` instead of `apriori` or `apriori.exe`, which are for Linux and Windows, respectively.)

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 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 `binarize()` included in the script `similarity.py`. (I.e., the script converts continuous data into a binary matrix based on dividing each attribute into two attributes by splitting it in above and below the median value). Save your binarized data into a text format that can be used by the apriori algorithm using the script `writeapriorifile.py`.

12.1.6 Find association rules using the `run_apriori.py` script specifying $minsupport \geq 30\%$ and $minconfidence \geq 60\%$. Try and interpret some of the associations with high support and confidence. Do these associations 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` as well as two additional entries to `attributeNamesBin` defining that the columns correspond to red and white wine.

12.2 Work on report three

For the remainder of this exercise work on report three due next week. In the association-mining section of the report investigate how attributes associate based on mining for association using the Apriori algorithm. You can use the function `binarize()` from `similarity.py` file to convert continuous data into a binary matrix based on one-out-of-K coding where each class correspond to a given percentile of the data. You can save the binary data matrix into a .txt file using `writeapriorifile.py` in order for the script `runApriori.py` to analyze the data (see also todays exercise using the wine data).

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.