Data Mining

Homework 3

Due: 3/5/2015, 23:59.

Instructions

You must hand in the homeworks electronically and before the due date and time.

Handing in: You must hand in the homeworks by the due date and time by an email to the instructor that will contain as attachment (not links!) a .zip or .tar.gz file with all your answers and subject

`[Data Mining class] Homework #`

where `#` is the homework number. After you submit, you will receive an acknowledgement email that your homework has been received and at what date and time. If you have not received an acknowledgement email within 1 day after you submit then contact the instructor.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

The solutions for the programming assignments must contain the source code, instructions to run it, and the output generated (to the screen or to files).

For information about collaboration, and about being late check the web page.

Most of the questions are not very hard but require time and thought. **You are advised to start as early as possible, to work in groups, and to ask the instructor in case of questions.**

Problem 1. For this exercise we will implement some versions of the A-priori algorithm.

1. Implement the simple A-priori algorithm in Python. For every round you have to read the file and you can keep in memory the frequent elements that you find.
   - You can store the itemsets during the computation in standard Python data structures (lists, sets, dictionaries) without doing the fancy stuff that we have seen.
   - The input format is one line per basket, and each line contains the IDs of the items in the basket separated by space.
   - You can assume that the IDs are $0, 1, 2, \ldots$.
   - At the end save the itemsets that you compute in a file, one line per itemset, each line containing the items in the itemset separated by space.
   - Report the total running time, and the number of itemsets found.
   - You will probably find very useful the `itertools` package.
2. Implement the simple, randomized algorithm of Section 6.4.1 of the book.
   - Read the file once, sample baskets into memory, and then compute the itemsets just by reading the memory.
   - Implement the two techniques of Section 6.4.2 to remove false positives and reduce false negatives.
   - At the end save the itemsets that you compute in a file, one line per itemset, each line containing the items in the itemset separated by space.

- Report the total running time, and the number of itemsets found, both before and after eliminating the false positives.

After making sure that your implementations are correct (by running them on some toy examples that you create) run them on two datasets:

1. First, on the file `http://aris.me/contents/teaching/data-mining-2015/homeworks/retail.dat.gz`, which contains the (anonymized) retail market basket data from an anonymous Belgian retail store. Use as threshold $t = 500$, and as sampling probability $p = 0.1$.

2. Then try them on the bigger dataset `http://fimi.ua.ac.be/data/webdocs.dat.gz`, which was built from a spidered collection of web html documents. Use as threshold $t = 500{,}000$ and as sampling probability $p = 0.0001$. This example shows why we often prefer approximate results: here the standard algorithm may be very slow (hours), whereas a good implementation (but without any smart techniques) of the randomized algorithm should be able to produce results in a few minutes.

For each of the datasets compare the results of the two algorithms. First write a small program to check that the randomized algorithm does not return itemsets that are not produced by the simple algorithm. Also compare the full results with the results that you obtain with the randomized algorithm if you set that threshold in the sampled set to (i) tp, and (ii) $0.9tp$. More specifically, report how many frequent itemsets does the randomized algorithm return for each of the two thresholds and how much time it needs.

Problem 2. Very often, when we search for frequent itemsets, we can be tricked: we may find items that are frequent even though the fact that they are frequent may be just because of chance and not because of any underlying reason. In this problem we will see an example of this situation.

Assume that we have n items, m baskets, and for every basket each item appears with probability p, independently of all the other items.

1. Consider an itemset of k items $\{i_1, i_2, \ldots, i_k\}$. Calculate what is the expected number of times that we will find the itemset in the m baskets.

2. Let's fix $n = 2000$, $m = 10^5$, and $p = 0.005$. How many times do we expect to see a particular item? How many times do we expect to see a particular pair of items?

3. Now perform some simulations using the values above and compute the frequency of the most frequent pair. Repeat them 10 times. How many times you did you find an item appearing more than 10% of its expectation? How many times did you find a pair that appears at least 5 times its expectation?

4. How do you explain the large difference between 2 and 3 in the case of pairs? And why do we get different results for single items and pairs?

Even though the data were generated completely at random, some pairs appear to be much more frequent than others. This example shows us that we should be careful not to draw fast conclusions. Generally, often we may think that there is some signal in our data, when in reality there is none.

Optional (extra credit): Propose and implement a way to deal with the problem above.