University of Ioannina,
Computer Science Department

Assignment 1

Questions 1-3 should be handed in at the beginning of the class on Wednesday April 4. Questions 4,5 should be turned in before 11:59pm on Friday April 6. For late submissions the late policy on the page of the course will be applied. The details for the turn-in will be announced on the page of the course.

Question 1 (15 units)

1. Let x and y be two vectors of Euclidean length (L_2 norm) equal to 1. What is the relationship between the Euclidean distance $d(x, y)$ and the cosine similarity $cos(x, y)$ of the two vectors?
2. Let $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots., y_n)$ be two vectors of size n. The sample Pearson correlation coefficient is defined as
$$\rho = \frac{\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_X)^2}\sqrt{\sum_{i=1}^{n}(y_i - \mu_Y)^2}}$$
where μ_X, μ_Y are the mean values of vectors X and Y respectively.
Describe how we can use the cosine similarity to compute the sample Pearson correlation coefficient.
3. Let $U = \{u_1, \dots, u_n\}$ be a set of n items, and let R be an ordering (ranking) of the items. Define a "reasonable" distance measure $d(R_1, R_2)$ between two rankings of the elements in U. Prove or disprove that your measure is a metric.

Question 2 (20 units)

A characteristic rule is a rule of the form $\{q\} \rightarrow \{p_1, p_2, \dots, p_n\}$, where the left-hand-side of the rule has a single item. Given an itemset of k items, we can generate k such characteristic rules. We define the measure ξ of an itemset as the minimum confidence c of any of these characteristic rules. That is,

$$\xi(p_1, p_2, \dots, p_k) = \min\{c(\{p_1\} \rightarrow \{p_2, \dots, p_k\}), c(\{p_2\} \rightarrow \{p_1, \dots, p_k\}), \dots, c(\{p_k\} \rightarrow \{p_1, \dots, p_{k-1}\})\}$$

Is the measure ξ monotone, anti-monotone, or non-monotone (neither monotone, nor anti-monotone)? Recall that for two sets X and Y, such that $X \subseteq Y$, a measure f is monotone if $f(X) \leq f(Y)$, and anti-monotone if $f(X) \geq f(Y)$.

Question 3 (20 units)

Consider the case where we have a dataset consisting of an ordered *sequence of events* (e.g., words in a document, or sequence of moves in a game). We want to find all frequent *subsequences* of events in the data that occur within a window W. For example, <A,B,C> is a subsequence of length 3 that occurs within a window of size W = 5, that contains the sequence <A,A,B,D,C>. The subsequence actually occurs twice, once starting at the first position, and once starting at the second position. The support of a subsequence is the number of windows of size W in the full sequence that contain the subsequence.

Describe how you will modify the apriori algorithm to compute all frequent subsequences with support at least *minsup*, for some given window size W. Be very clear on the issue of candidate generation, candidate pruning, and frequency calculation. You can modify the algorithm however you wish, as long as your algorithm is correct and efficient.

Apply your algorithm on the sequence ABCACDBAAC with W=4 and *minsup* = 2. Report the output of all the intermediate steps of the algorithm: candidates generated, candidate pruning, frequent sequences, in the order in which you create them. Indicate if a sequence is the result of merging two smaller subsequences.

Question 4 (15 units)

The goal of this exercise is to familiarize yourselves with the WEKA data mining software. You can download WEKA from http://www.cs.waikato.ac.nz/ml/weka/ . You can find information about how to set up and run WEKA on the same site, and on the page of the course. You will use the "Congressional Voting Records" dataset (downloaded from the UCI repository), which consists of the congressional votes of 465 US congress members on 16 different subjects, plus a class attribute (democrat, republican). The ARFF file (used for input to WEKA) can be found on the page of the course.

Using support threshold 0.1, produce all frequent itemsets in the data. Also produce the 10 rules with the highest confidence and the 10 rules with the highest lift (interest). Turn in a file with the results, and the parameters that you set in WEKA.

Question 5 (30 units)

In this exercise you will implement min hashing and Locality Sensitive Hashing. You will test your implementation on the MovieLens 100k dataset which consists of a set of 943 users that have rated 1682 movies. . You can download the data from http://www.grouplens.org/node/73. Read the Readme file for details about the data, and process it as you need. For this exercise, we only care about the *set of movies* that a user has rated, and not the ratings. We want to compute the Jaccard similarity between the users

Compute the exact Jaccard similarity for all pairs of users and output the pairs of users that have similarity at least 0.5. Then compute the min-hash signatures for the users, and compute the approximate Jaccard similarity as we described in class. Use 50, 100, and 200 hash functions. For each value, output the pairs that have estimated similarity at least 0.5, and report the number of false positives and false negatives that you obtain. For the false positives and negatives, report the averages for 5 different runs.

Next, break up the signature table into b bands with r hash functions per band, as discussed in class, and implement Locality Sensitive Hashing. The goal is to find candidate pairs with similarity at least 0.6. Experiment with r=5, b =10 for the table with the 50 hash functions, r=10, b=10 for the table with the 100 hash functions, and r=20, b= 10 for the table with the 200 hash functions. Report the number of false positives and false negatives taking the average over 5 runs. How do these numbers change if we want similarity at least 0.8?

You should turn in your code, the input file that you used, and the output files that you produced for the different runs. Also turn-in a file with the average numbers, and your observations.

Technical details

1. For pre-processing the data, some unix commands that you may find useful are the following:
    a. cut: allows you to get specific columns from delimited data
    b. sort: sorts the rows of a file in lexicographic order, –n for numeric
    c. uniq: merges consecutive rows of a file that are identical.
2. Use the following hash function for the signatures:

    Select a large enough prime number R (e.g., R = 131071);

    Select a,b,c, random numbers in the interval [0,R]

    h(x) = (a*(x>>4) +b*x +c)%R;
3. For the implementation of LSH you do not need to implement a hash table or a list. Use existing implementations that come with the language (e.g., in C++ there is the STL implementation, check the web page of the object oriented programming course for more details).