CSCI-B565: Data Mining

Homework Assignment # 4

Assigned: 03/24/2018 *Due: 04/09/2018, 11:59pm, through Canvas*

Three problems, 145 points in total. Good luck!
Prof. Predrag Radivojac, Indiana University, Bloomington

Problem 1. (20 points) Let $\{d(x_i, x_j)\}_{ij}$ be a set of $\binom{n}{2}$ Euclidean distances between pairs of n distinct objects from some space $\mathcal{X}$. Suppose the data set of objects is partitioned into m groups, where $\{c_j\}_{j=1}^{m}$ is a set of m centroids for each (non-overlapping) subset of objects j; that is, for each subgroup of objects $j \in \{1, 2 \dots m\}$, c_j is their centroid. Now, if two of the m groups of objects are to be merged to minimize the sum of squared errors between each data point and its respective centroid, show how you will compute which two clusters to merge out of $\binom{m}{2}$ pairs in total. Note that you are only allowed to work with distances between these points because you are not given the original objects x_i. You are not allowed to use any embedding techniques such as multidimensional scaling, to map these points into a Euclidean space. Prove any complex statements in your algorithm.

Problem 2. (75 points) Apriori algorithm. Implement the *Apriori algorithm* by first determining frequent itemsets and then proceeding to identify association rules. Consider that the input to your program is a sparse matrix where the rows are transactions and columns are items. Each value in your matrix is a binary variable from $\{0, 1\}$ that indicates presence of an item in the transaction.

a) (20 points) Implement both $\mathbf{F}_{k-1} \times \mathbf{F}_1$ and $\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ methods. Allow in your code to track the number of generated candidate itemsets as well as the total number of frequent itemsets.

b) (10 points) Use three data sets from the UCI Machine learning repository to test your algorithms. The data sets should contain at least 1000 examples, and at least one data set should contain 10,000 examples or more. You can convert any classification or regression data set into a set of transactions and you are allowed to discretize all numerical features into two or more categorical features. Compare these two candidate generation methods on each of the three data sets for three different meaningful levels of the minimum support threshold (the thresholds should allow you to properly compare different methods and make useful conclusions). Provide the numbers of candidate itemsets considered in a table and discuss the observed savings that one of these methods achieves.

c) (10 points) Enumerate the number of frequent closed itemsets as well as maximal frequent itemsets on each of your data sets for each of the minimum support thresholds from the previous question. Compare those numbers with the numbers of frequent itemsets.

d) (15 points) Implement confidence-based pruning to enumerate all association rules for a given set of frequent itemsets. Use the previous data sets, with three levels of support and three levels of confidence to quantify the savings in the number of generated confident rules compared to the brute-force method.

e) (10 points) For each data set and each minimum support threshold, select three confidence levels for which you will generate association rules. Identify top 5 association rules for each combination of support and confidence thresholds and discuss them (i.e., comment on their quality or peculiarity). Select data sets where you can more easily provide meaningful comments regarding the validity of rules.

f) (10 points) Instead of confidence, use *lift* as your measure of rule interestingness. Identify top 5 rules for each of the previous situations and discuss the relationship between confidence and lift.

No specialized libraries are allowed for this task. Make sure that you code runs and submit all the code you used in this task, including the code that converts data sets from UCI Machine Learning repository into a transaction data set. Do not include raw data sets in your supplement; however, do provide links to the data sets you used such that your code can be run independently and its performance can be verified.

Problem 3. (50 points) Building a recommendation system. In this exercise you will test different distance functions for the movie recommendation systems. First, familiarize yourself with the MovieLens data sets that are available at

`http://grouplens.org/datasets/movielens/`

Next, design and evaluate your own recommendation system based on the following principles:

- For each user i and each movie j they did not see, find top k most similar users to i who have seen j and then use them to infer the user i's rating on movie j. Handle all exceptions in a reasonable way and report your strategy if you did so; e.g., if you cannot find k users for some movie j, then take all users who have seen it.

- Test the performance of your system using cross-validation. For each data set, the MovieLens database already provides a split of the initial data set into $N = 5$ folds. This means you will run your algorithm N times; in each step, use the training partition to make predictions for each user on all terms rated in the test partition (by that user). When you complete all N iterations, you will have a large number of user-movie pairs from the 5 test partitions on which you can evaluate the performance of your system.

- Measure the performance of your recommendation system using the mean absolute difference (MAD); that is,

$$\text{MAD} = \frac{1}{\sum_i \sum_j r_{ij}} \cdot \sum_i \sum_j r_{ij} \cdot |p_{ij} - t_{ij}|$$

  where p_{ij} is the predicted rating of user i on movie j, t_{ij} is the true rating available in the test partition, and r_{ij} is the indicator variable of the availability of rating for the pair (i, j). That is, $r_{ij} = 1$ if the rating for (i, j) is available; otherwise, $r_{ij} = 0$.

- Compare all your algorithms with a simple algorithm that gives each user-movie pair a rating that equals the average score over all users who rated the movie. Note that here each user receives the same rating (prediction) for a particular movie. Ideally, your algorithm will outperform this terribly naive scheme.

Your task in this exercise is to train and evaluate several recommendation systems.

a) (15 points) Use the "100K Dataset" to evaluate three different distance functions of your choice (must include the Euclidean distance) using the entire vectors of ratings over all movies. Calculate the performance and find the best k from a hand-selected set of 3-5 values (your choice) for each specific case.

b) (15 points) Continue with the "100K Dataset", but modify your distance metrics appropriately to incorporate other information such as user's gender, movie genre, etc. You should also at this stage change your choice of distances into the ones you believe might perform better. For each new function you tested, explain the rationale for choosing it. It is not required, but you can also attempt one modification of the first principle of the recommendation system stated above (the algorithm itself).

c) (15 points) Switch to the "10M Dataset" and evaluate the performance of the top three algorithms you devised in the previous steps (this includes variations over distance functions and parameters). Note that here you will need to use the script to generate training and test partitions. Discuss your observations and findings. Note that this part of the assignment might take long time to run and that you need to think of a good strategy early on to prioritize your work in this assignment.

d) (5 points) Briefly comment on what might be a good next set of steps to improve this recommendation system.

We have created directories `/l/b565/ml-100k` and `/l/b565/ml-10M100K` on the Hulk server and ran the .sh script in the latter case. These files are readable by anyone. Therefore, please avoid creating copies of these files on the Indiana University servers. To log on to Hulk, go to

`hulk.soic.indiana.edu`

and use your university ID and password.

Homework Directions and Policies

Submit a single package containing all answers, results and code. You submission package should be compressed and named firstnamelastname.zip (e.g., predragradivojac.zip). In your package there should be a single pdf file named main.pdf that will contain answers to all questions, all figures, and all relevant results. Your solutions and answers must be typed[1] and make sure that you type your name and IU username (email) at the beginning of the file. The rest of the package should contain all code that you used. The code should be properly organized in folders and subfolders, one for each question or problem. All code, if applicable, should be turned in when you submit your assignment as it may be necessary to demo your programs to the associate instructors. Use Matlab, Python, R, or C/C++.

Unless there are legitimate circumstances, late assignments will be accepted up to 5 days after the due date and graded using the following rules:

on time: your score $\times$ 1

1 day late: your score $\times$ 0.9

2 days late: your score $\times$ 0.7

3 days late: your score $\times$ 0.5

4 days late: your score $\times$ 0.3

5 days late: your score $\times$ 0.1

For example, this means that if you submit 3 days late and get 80 points for your answers, your total number of points will be $80 \times 0.5 = 40$ points.

All assignments are individual, except when collaboration is explicitly allowed. All the sources used for problem solution must be acknowledged; e.g., web sites, books, research papers, personal communication with people, etc. Academic honesty is taken seriously! For detailed information see Indiana University Code of Student Rights, Responsibilities, and Conduct.

[1] We recommend Latex; in particular, TexShop-MacTeX combination for a Mac and TeXnicCenter-MiKTex combination on Windows. An easy way to start with Latex is to use the freely available Lyx. You can also use Microsoft Word or other programs that can display formulas professionally.