CS/INFO 5304 Assignment 2: Recommender Systems

Credit: 35 points
Grade: 20% of final grade
Submission: Submit all code and results asked within the individual sections *and* the results specified
Tip: Read the document fully before you begin working on the assignment
Due date: April 17th, 11:59PM

The goal of the assignment is to get familiar with different types of recommender systems. Specifically, we are going to build a system that can recommend TV shows to users.

You may use Python for this assignment, and **it is not necessary to use Pyspark.**

Dataset:

The dataset that we will be using contains information about TV shows. More precisely, for 9985 users and 563 popular TV shows, we know if a given user watched a given show over a 3 month period.

The folder contains:

• user-shows.txt - This is the ratings matrix R, where each row corresponds to a user and each column corresponds to a TV show. Rij = 1 if user i watched the show j over a period of three months. The columns are separated by a space.

• shows.txt - This is a file containing the titles of the TV shows, in the same order as the columns of R.

The dataset can be found here.

https://drive.google.com/file/d/1WH6YCcWsm09oG3tYxokyTmVig5EGwM9H/view?usp=sharing

Questions:

In this assignment we are going to implement three types of recommender systems namely

- User - User recommender system
- Item – Item recommender system
- Latent factor model recommender system

We are then going to compare the results of these systems for the 500th user of the dataset. Let's call him Alex. In order to do so, we have erased the first 100 entries of Alex's row in the matrix, and replaced them by 0s. This means that we don't know which of the first 100 shows Alex has watched. Based on Alex's behavior on the other shows, you need to give Alex recommendations on the first 100 shows. We will then see if our recommendations match what Alex had in fact watched.

Part A: user – user recommender system [10 points]

In a user-user recommender system, you need to find users who have given a similar rating to the TV-shows. Then among these users, you can recommend the top rated items.

To implement this, you need to compute the following

for all Tv-shows t, compute $r_{Alex,t} = \Sigma_{x \in users}$ cos-sim(x, Alex)$\cdot R_{xt}$
where cos-sim(x,Alex) is the cosine similarity of other users with Alex and R is the ratings matrix. In the above equation you are first finding the similarity between users and then multiplying it with their product rating for each item. So the tv-shows that have higher $r_{Alex,t}$ will be the shows that are rated high by the users similar to Alex.

Let S denote the set of the first 100 shows (the first 100 columns of the matrix). From all the TV shows in S, which are the five that have the highest similarity scores ($r_{Alex,t}$) for Alex? What are their similarity scores? In case of ties between two shows, choose the one with a smaller index. Do not write the index of the TV shows, write their names using the file shows.txt.

Part B: item – item recommender system [10 points]

In a item-item recommender system, you need to find items that have similar ratings and recommend it to Alex

To implement this, you need to compute the following

for all Tv-shows t, compute $r_{Alex,t} = \Sigma_{x \in items} R_{Alex,x} \cdot$ cos-sim(x, t)

where R is the ratings matrix and cos-sim(x,t) is the cosine-similarity of each pair of TV shows. Here you are finding similar items and then multiplying it with Alex's ratings for items. So the shows that are similar to the shows already watched by Alex will have the higher rating r

From all the TV shows in S (first 100 shows), which are the five that have the highest similarity scores for Alex? In case of ties between two shows, choose the one with a smaller index. Again, hand in the names of the shows and their similarity score.

Part C: Latent hidden model recommender system [15 points]

Latent model recommender system is the most popular type of recommender system in the market today. Here we perform a matrix factorization of the ratings matrix R into two matrices U and V where U is considered as the user features matrix and V is the movie features matrix. Note that the features are 'hidden' and need not be understandable to users. Hence the name latent hidden model. (refer slides for more information)

The latent model can be implemented by performing a singular value decomposition (SVD) that factors the matrix into three matrices

$$R = U \Sigma V^T$$

where R is user ratings matrix, U is the user "features" matrix, Σ is the diagonal matrix of singular values (essentially weights), and V^T is the movie "features" matrix. U and V^T are

orthogonal, and represent different things. U represents how much users “like” each feature and V^T represents how relevant each feature is to each movie.
To get the lower rank approximation, we take these matrices and keep only the top k features, which we think of as the k most important underlying taste and preference vectors.

(Hint – You can use scipy library to perform svd)

With k set to 10 first perform SVD to identify the U and V matrices. You can then multiply the matrices to estimate the following

$$R^* = U \Sigma V^T$$

From the R* matrix, select the top 5 TV-shows for Alex in S (first 100 shows). In case of ties between two shows, choose the one with a smaller index. Again, hand in the names of the shows and their similarity score.

To submit:

You need to submit a zip file containing the following in Canvas

a) A text file with the answers to Part A, B and C
b) Source code files