CS 5683: Big Data Analytics

Assignment-5: Collaborative Filtering

Total Points: 30 (3% towards final)

Due date: Oct 27, 2021 at 11:59pm

This assignment is a preliminary for Project-4. In this assignment, we will experiment with a simple item-item collaborative filtering discussed in our class. We will evaluate the performance of the algorithm with Root Mean Squared Error (RMSE). We will utilize this performance for comparing models that we develop in Project-4.

Dataset: We will use the openly available movie ratings data in this project (Source: https://grouplens.org/datasets/movielens/100k/). We have processed the data and made training and test data samples available. Both training and test data have columns: '*user_id*', '*item_id*', '*rating*', and '*movie_name*'. *The 'rating' feature in the test_dataset should be used only for model evaluation*. In other words, you must not use the 'rating' feature for similarity measure in the Item-Item Collaborative Filtering.

Consider the training dataset as a matrix R of ratings. The element R_{xi} of this matrix corresponds to the rating given by user x to movie i. The size of R is m x n, where m is the number of users, and n is the number of movies. Most of the elements of the matrix R are unknown because each user can only rate a few movies.

Item-item Collaborative Filtering:

Unlike latent factor models, we do not require to optimize any error functions in item-item collaborative filtering. Instead, we will read the test data (of course, without '*rating*') and give predictions directly. Our task includes (i) to set the similarity metric, (ii) to calculate similarities of multiple movies, and (iii) to predict the rating of the given movie ***i*** for given user ***x***. Predictions using the item-item collaborative filtering can be calculated using **Eq. 1**:

$$R_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} * R_{xj}}{\sum_{j \in N(i;x)} s_{ij}} \qquad \textit{Eq. 1}$$

where N is a set of movies that are similar to the movie ***i*** and also rated by the user ***x***, s_{ij} is the similarity score of items ***i*** and ***j***. We compute similarity score s_{ij} of two movies ***i*** and ***j*** with *Cosine similarity* given in **Eq. 2**:

$$sim(i,j) = \frac{\sum_y^U R_{yi} \cdot R_{yj}}{\sqrt{\sum_y^U R_{yi}^2} \sqrt{\sum_y^U R_{yj}^2}} \qquad \textit{Eq. 2}$$

where U is the set of all users who have rated movies ***i*** and ***j***

NOTE-1: You do not need to calculate $sim(i, j)$ for all movie pairs in this project. You only need to find similarity of movies that were rated by user ***x***. You can pre-compute this similarity by consolidating a list of movies from the test data

NOTE-2: Consider movies that are at least 50% similar for the set ***N*** in **Eq. 1**. You can fine tune the parameter ***N*** for both cosine similarity and adjusted cosine similarity to improve the model performance

Model Evaluation:

Implement the following steps to do evaluation for all 3 models:

1. Read the test dataset and hide the '*rating*' column *[**NOTE:** We provided this dataset: **test_dataset_without_rating.csv**]*
2. Predict the '*rating*' value using models discussed above
3. Compare '*actual_rating*' and '*predicted_rating*' with Root Mean Squared Error (RMSE). Code snippet to calculate RMSE is given below:

```
from sklearn.metrics import mean_squared_error

from math import sqrt

def RMSE(y_actual, y_predicted):
rms = mean_squared_error(y_actual, y_predicted, squared=
False)
return round(rms,4)
```

Bonus task:

Implement two other recommender models: *Global baseline estimates* and *Collaborative filtering with Adjusted Cosine Similarity*.

1. *Global baseline estimates*: A simple strategy to predict movie ratings is just with global estimate given in **Eq. 3:**

$$R_{xi} = \mu + R_x^* + R_i^* \qquad \textit{Eq. 3}$$

where μ = Overall mean movie rating
R_x^* = Average rating of user **x** - μ
R_i^* = Average rating of item **i** - μ

2. *Adjusted cosine similarity*: As discussed in class, cosine similarity works, but it does not consider different rating schemes of different users. To avoid this scenario, we modify the cosine similarity score with **Eq. 3**:

$$sim(i,j) = \frac{\sum_y^U \left(R_{yi} - R_y^*\right)\left(R_{yj} - R_y^*\right)}{\sqrt{\sum_y^U \left(R_{yi} - R_y^*\right)^2}\sqrt{\sum_y^U \left(r_{yj} - R_y^*\right)^2}} \qquad \textit{Eq. 4}$$

where R_y^* is the average rating of user ***y***.

Submission requirements:

1. Students can utilize Python programming
2. Programs should be well documented – the grader should understand program modules clearly with your documentation
3. Submit only the following one python file for this assignment [YES, you can submit a notebook (.ipynb) file also]
4. What to output in each task?
    a. RMSE for the test data using cosine similarity
    b. If you are trying for bonus points, output RMSE of global baseline, collaborative filtering with cosine similarity, and collaborative filtering with adjusted cosine similarity. Compare all results and give resoning

Grading Rubric:

1. Collaborative Filtering – Total: 30 points
    a. Cosine similarity: 15 points
    b. Results: 10 points
    c. Documentation: 5 points
2. Bonus: 15 points
    a. Global baseline implementation: 5 points
    b. Collaborative filtering with Adjusted Cosine Similarity: 3 points
    c. Compare and contrast all results: 2 points