Homework 4 - CS584

Hoang-Dung Bui
Team Name: bui1720
Mason ID: G01301478
Rank:
Highest Public Score 2.48

1 Introduction

In this homework, a recommendation system was developed to calculate the rating for a list of movies with userID. There are two systems developed: (1) Item-based K-NN and (2) User-Item Based Matrix Factorization. The first one provided a faster running with higher accuracy (2.48). Meanwhile, the other one took a long time to process, then provided a worse result.

2 Item-Based K-NN

Model's Selection:

- Cosine Similarity is used to calculate the distance. The top n items will be selected to calculate the rating for the user and item.
- To improve the computation efficiency of the K-NN model, the package NearestNeighbors of sklearn was used. It sped up the processing time much more than my built code from scratch.
- The data from the train.dat file was converted into user-item matrix, which was filtered the movies and users which had more then the rating threshold, respectively.

Code's Explanation: A class of the k-nn model was developed, which processed the data file, rating the movies for the users, and save the results into a file. It consists of the functions: *preprocess_data*, *_rating*, and *make_predictions*

- *preprocess_data*: It used the provided link to gain the data (train and test). The test data was in form of pandas DataFrame. The train data was filtered to remove the un-popular movies and inactive users. The data frame was then transformed into user-item matrix by *pivot* function of pandas package, and then sparse matrix by *csr_matrix* The output of this functions were the sparse matrix, user-item matrix and the test DataFrame.
- *_rating*: Its inputs were: the k-nn model, train user-item matrix, sparse matrix, and a pandas DataFrame of testing data. First the sparse matrix was inputted into the k-nn model; then a pair of (userID and movieID) from the rating_movies DataFrame was inputted to the k-nn model to determine n closest neighbors. From n neighbors, the prediction value was then calculated for the (userID and movieID) pair, then saved into a list. Due to an un-explainable reason, the rating was negative sometimes. All the negative results were assigned to zero.
- *make_predictions*: This function called the two previous functions, and pass the list of rating results to the main function.
- *main* function: does nothing else than passing the arguments (data files' name and model's parameters), and save the rating result list into the output file.

Experiment Results: The neighbor number of k-nn was set to 10. The thresholds for active user and popular movie were varied (2, 4, 6, and 10) during the experiment. For the thresholds of 2 and 4, the system make a good prediction of score 2.48. As increasing the threshold to 6 and 10, the prediction scores were reduced into 2.47 and 2.46, respectively.

3 User-Item Based Matrix Factorization

Model's Selection:

- The latent variables need to be selected (the model was tested with 3 and 5).

Code's Explanation: A class of the matrix factorization model was developed, which processed the data file, rating the movies for the users, and save the results into a file. It consists of the functions: *preprocess_data*, *matrix_factorization*, *_calculate_error*, *_update_P_Q*, and *scoring*

- *preprocess_data*: It used the provided link to gain the data (train and test). The test data was in form of pandas DataFrame. The train data was transformed into user-item matrix by *pivot* function of pandas package. The output of this functions were the user-item matrix and the test DataFrame.
- *matrix_factorization*: This function generated the factorized matrices and then updated them in *step* iteration.
- *_update_P_Q*: This private function updated the factorized matrices in each iteration. This function could be run parallel.
- *_calculate_error*: a private function in the recommender_MF class, which calculated error of the factorized matrices each time invoked. This function could be run parallel.
- *scoring*: This function received the user-item matrix, test dataFrame, and the factorized matrix. For each pair of (userID and movieID), it will get the corresponding value in the factorize matrix (R_new). The values were then saved into a list, and passed back the main function.
- *main* function: does nothing else than passing the arguments (data files' name and model's parameters), calling the recommender_MF class, and save the rating result list into the output file.

Experiment Results: As the latent variable was assigned to 3 and 5, the model still provided very bad result (the score of 0.8). There must be some errors in the program, however, I have not found them.

4 Conclusion

In this homework, two recommendation systems were developed to score the rate points for particular users and movies. The Item-Based K-NN worked well on the dataset and consumed significant short computation time. Meanwhile, the User-Item Based Matrix Factorization system did not rate well for the dataset. Moreover, it needed much longer time than the first system. I think there is error in program, and I need more time to find it out.