Jeremiah Akinbohun

Miner username: joabrb22

CS 484-001

**HW4 - Recommendation System**

**Rank: 3**

**RMSE Score: 3.0**

**Best Runtime: ~21 mins**

**Special Instruction for Running Code:** Runtime for this code is long due to data frame and matrix operations, please see runtime above.

**My Approach**

I chose to use matrix factorization in order to fill in the missing user ratings. I began by reading in the data and creating a utility matrix from the users, the movies, and the known ratings each user gave to the movies that they rated. For movies that they did not rate, I filled in those missing ratings with a 0. The user’s bias must be accounted for at this point, i.e., some users are more likely to rate the movies they do rate high or low. Finding this mean for each user’s ratings also allows for filling in matrix with information that can be used to help with matrix factorization as well. I did encounter an issue here. When applying the pandas mean function on the rows of the utility data frame, the missing values, filled in with zeros, were counted towards calculation of the mean. Filling in the missing values takes a significant amount of my program’s runtime. I created a function that derived each user’s mean rating from the sum of the ratings they had given divided by the total number of movies that that user had rated. During the course of running this function, I found that some users had no ratings. I assume that these users are new and have yet to rate any movies, so I had to account for this case as well. Computing the mean user ratings also took a significant amount of run time. Given the sparse nature of the matrix, svds works well to generate predictions with a minimal amount of factors/information. I set svds to calculate the ratings vectors for each user by using a linear combination of k = 50 vectors to build back the utility matrix with the missing ratings filled in, thus giving a good approximation of what the data is supposed to look like. My understanding of k is that it is an approximation of the best k # of users that represent the data. Like taking the first most significant layer of pixels from a black and white picture, and then progressively adding more pixels with each rank until the picture begins to reform. With all of the ratings either known or approximated, I iterate through the test data and find the predicted movie rating for each user.

**Methodology**

With the inclusion of many datasets for this assignment, I began by examining the data to understand the high-level relationships between them. I noticed that users had ratings for some movies and not all, that each movie had accompanying genre, actor, and tag information, and that those tags had weight. Initially, I considered using a nearest neighbors approach. This would have involved creating a similarity matrix for users and movies. From there, in order to determine what rating a user would give a movie, I would search for k-nearest neighbors to the user and average their ratings for the movie whose rating was to be guessed. If a movie had no ratings, or the KNNs had not rated that movie then the algorithm would select the next most similar movie from the movie similarity matrix and the k-nearest neighbors would again be checked for their ratings. Although I viewed this as being simple, the matter of dealing with not having enough KNNs for the test data movie selected would require some improvisation. Additionally, from my research, this method of a memory-based approach does not work well with very sparse data and data with high dimensionality and so I chose to not develop it. I considered a content-based approach as well, but concerns about handling new users, who would have no user profile, too much similarity among movies not allowing for choices outside of firmly set groupings, and the case of a movie not having enough tags or other information made me eschew using such an approach. The collaborative filtering approach was the most intuitive. To me, it still relied on similarity but had the advantage of using information I could not ascertain on the surface to provide hidden insights, and that is why I chose to use a model-based approach.

Another significant reason I chose the model-based approach was it was simpler to deal with the ratings than generate data frames and matrices from the wide breadth of data made available. I used only the movie genre data set to find the unique movie IDs, the training data set to find the unique user IDs and pulled the ratings into the utility matrix I created.