MOVIE RECOMMENDER SYSTEM

CS6375 Project: Final Report
May 2018

Pramod Hegde
pxh170330@utdallas.edu

Ram Gopal V
pxv172130@utdallas.edu

ABSTRACT:

Collaborative filtering(CF) is the most common practice in the field of recommendations systems. In this project, we implemented the two popular methods of CF known as neighborhood problem and latent factor approaches on Movie Lens dataset (100K). In neighborhood problem, we worked on both user-user and item-item similarities for predicting rating. Next in latent factor we used singular value decomposition(SVD) to find the users similarity in movie concept scape. Item-item in neighborhood and SVD resulted in less Root Mean Square Error.

1. INTRODUCTION:

Recommendation system attempts to predict the rating and narrows down the selection criterion for a user based on his past rating and activities. Recommender system has been gaining increasing popularity in the recent years in fields of music, videos, web searches to name a few. There are many systems that keep track of the user's activities and attempts to suggest new items based on the collected information. A popular example would be Netflix recommendation system which lists the popular TV shows and the ones that a user would like based on the interest he expresses while setting up the account. By advancement in recommendation system users always wants to get a good recommendation or they discontinue to use the feature. This has led for a major improvement in the techniques that are being used for recommendation system.
The two most popular approaches used in the recommendation system are content-based filtering and collaborative filtering. In content-based filtering, items are recommended based on comparison between the contents of item and users. Set of descriptors makes up the contents and it serves as a profile to recommend a user. This approach can be used even if we have a single user. In collaborative filtering, we try to group similar users and attempt to recommend items to a user based on the items liked by the users in that group, one problem with this technique is cold start problem we cannot recommend items until we have some amount of user data.

2. ALGORITHMS

For this project, we focused on collaborative filtering approach. We worked on three algorithms involved in collaborative filtering.

2.1 Global Baseline Recommender:

Most of the collaborative filtering starts with a common baseline model. There are two types of effects observed in the baseline predictor model. First, the popularity of an item may change over time this is treated as item bias b_i . Second, every user is unique, and their interest may change over time hence their ratings may change this is treated as user bias b_u. So, the baseline model for our model is taken in this form.

$$b_{ui} = \mu + b_i + b_u \qquad (1)$$

µ is the overall average rating.

$$\min_{b_*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i)^2 + \lambda(\sum_u b_u^2 + \sum_i b_i^2) \quad (2)$$

In the first term of eq (2) tries to find the parameters and second regularizing term tries to avoid overfitting by penalizing the item and user bias

For an efficient method which learns model parameter we can use the following techniques. For each given rating r_{ui}, a prediction ($\hat{r}_{ui}$) is made and associated prediction error $e_{ui} = r_{ui} - \hat{r}_{ui}$ is computed and for every given training case we update the parameters using the following equation:

$$b_u \leftarrow b_u + \gamma\,(e_{ui} - \lambda\, b_u)$$
$$b_i \leftarrow b_i + \gamma\,(e_{ui} - \lambda\, b_i) \qquad (3)$$

The above steps are repeated for few iterations so that the bias reaches the global minima using gradient descent. Finally, we will use the updated b_u and b_i 's to predict the rating and to check the accuracy we will calculate the RMSE using the formula (11)

2.2 Local Neighborhood (K Nearest Neighbor(KNN)):

In KNN to predict a point (u, m), we are going to compute the K most similar points and find the average of the ratings to find the predicted rating.
The approaches we used for the project are user-based and item-based K-nearest neighbor algorithm,

Pearson Correlation:
There are many ways to form a similarity matrix, one that has been mostly used in collaborative filtering is a measure based on the correlation between movie ratings.

$$\text{sim (i, j)} = \text{PC (i, j)} = \frac{\sum_{u\in U_{ij}} (r_{ui}-\overline{r}_i)(r_{uj}-\overline{r}_j)}{\sqrt{\sum_{u\in U_{ij}} (r_{ui}-\overline{r}_i)^2 \sum_{u\in U_{ij}} (r_{uj}-\overline{r}_j)^2}} \quad (4)$$

KNN finds the K-nearest neighbors using the above formula and uses the weighted means to predict the rating. The formula used by the KNN is.

$$r_{m,n} = \frac{\sum_{j\in N_u^{K_m}} sim(m, j) r_{ju}}{\sum_{j\in N_u^{K_m}} |sim(m, j)|} \quad (5)$$

where $N_u^{K_m}$ = {j: j belongs to K similar movies to m and user u has rated j}.

User-User Nearest Neighbor:
To implement this, for any given user we will calculate the similarity with all other users using Pearson correlation. Then we select the k-nearest neighbors based on the similarity matrix that was calculated.

Item-Item nearest neighbor:
In this method, for any given movie m we calculate the similarity with all other movies. To determine which movies are like movie m, we need to form a similarity matrix. Then use the weighted means to predict the rating.

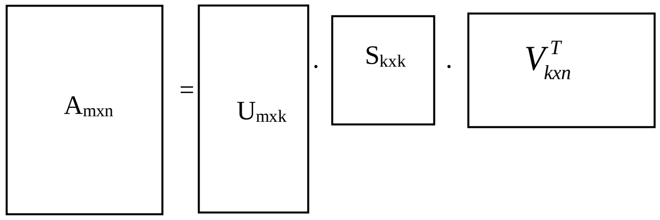
Item-Item and User-User nearest neighbor (baseline model included):
The previous technique is modified by including baseline model. Only k top users with similarity are picked. Their corresponding deviation in rating and the weighted average of similarity level is calculated.

$$r_{xi} = b_{xi} + \frac{\sum_{j\in N(i;x)} s_{ij}.(r_{xj} - b_{xj})}{\sum_{j\in N(i;x)} s_{ij}} \quad (6)$$

2.3 SINGULAR VALUE DECOMPOSITION

SVD is used to represent the information in minimal coordinates, it is a matrix factorization technique. It finds the top k features that are important to represent the information. We can use two matrices U (user features), V (item features) to generate a matrix A of m x n in which m rows are items and n columns are users S is a diagonal matrix with r nonzero entries each representing the importance of each feature.

$$A_{mxn} = U_{mxk} S_{kxk} V_{kxn}^T$$



Each item is represented by a vector q_i. and the user is represented by a vector p_u where p_u is the amount of interest a user has in items and q_i is the factors possessed by the item. To get the interaction between a user and an item we need use the dot product of $q_i^T p_u$

The SVD always tries to find the best U and V for this purpose the function makes use of only observed values which causes overfitting. To overcome the overfitting, we will make use of regularization parameter λ, which controls the model parameters

$$parametrs = \frac{1}{\lambda} \quad (9)$$

By introducing the λ. The objective function becomes as follows

$$\min_{p^*,q^*} \sum_{(u,i)\in K} \left(r_{ui} - p_u q_i^T\right)^2 + \lambda\left(\|p_u\|^2 + \|q_i\|^2\right) \quad (10)$$

```
Algorithm for SVD
Input:
user-item rating $U_{ij}$ matrix and the dimension of
P and Q.
Output:
Root Mean Square Error (RMSE) of the test data
set.
Method:
Step1: Initialize the user feature matrix p and the
item feature matrix q. Let the values $p_{ij}$ and $q_{ij}$ be
0.1 random (0,1)
Step 2: Calculate the predicted rating values user
for item i using $\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$
Step 3: Calculate $e_{ui}$ according to the formula of
$e_{ui} = r_{ui} - \hat{r}_{ui}$
Step 4: update
$b_u \leftarrow b_u + \gamma ( e_{ui} - \lambda b_u)$
$b_i \leftarrow b_i + \gamma ( e_{ui} - \lambda b_i)$
$p_u \leftarrow p_u + \gamma (e_{ui} q_i - \lambda p_u)$
$q_i \leftarrow q_i + \gamma (e_{ui} p_u - \lambda q_i)$
Step 5: Repeat to step 3 until we predict all the
ratings.
Step 6: Calculate $\hat{r}_{ui}$ of each rating and then RMSE
in the test dataset
```

3. EXPERIMENTAL EVALUATION:

3.1 DATA SET:

Data used in this project is set of CSV files from Movie Lens dataset. The data consists of 9126 number of movies and ratings of 100005 from 671 different users with all the users have rated at least 20 movies. No demographic information was included. We represented data in form of utility matrix containing user-item pair and ignored extra information like genre and tags. We split the data into 80% and 20% for training and testing purpose.

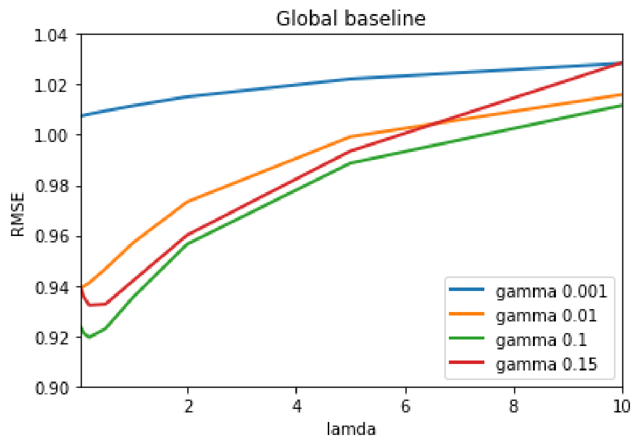
3.2 Error Metric:

The error matric we are using for the project is the root mean squared error one of the most used metric while evaluating the systems. RMSE says how close is prediction compared to the actual rating and it is given by the rule

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in TestSet} (r_{ui} - \hat{r}_{ui})^2}{|TestSet|}} \qquad (11)$$

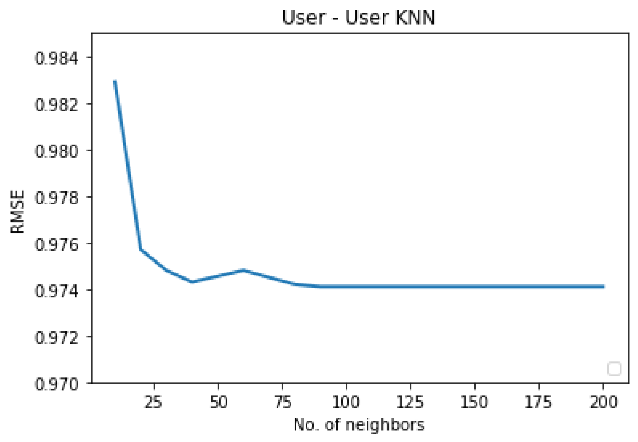
Where $\hat{r}_{ui}$ is predicted rating and r_{ui} is the target rating.

Results:



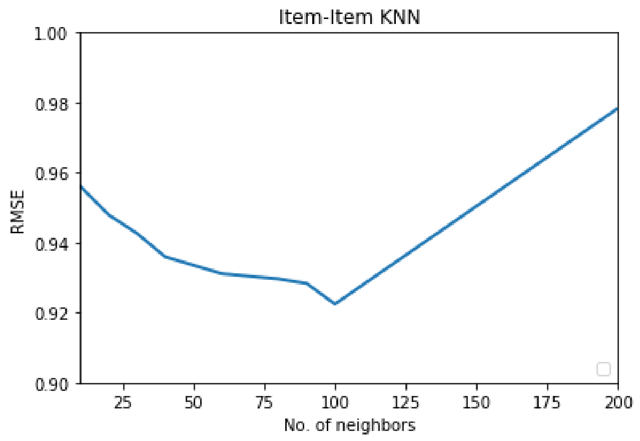
(a)

The above plot shows the variation of lambda. As we increase the lambda the model underfits and results in high RMSE because it controls the user, item bias.



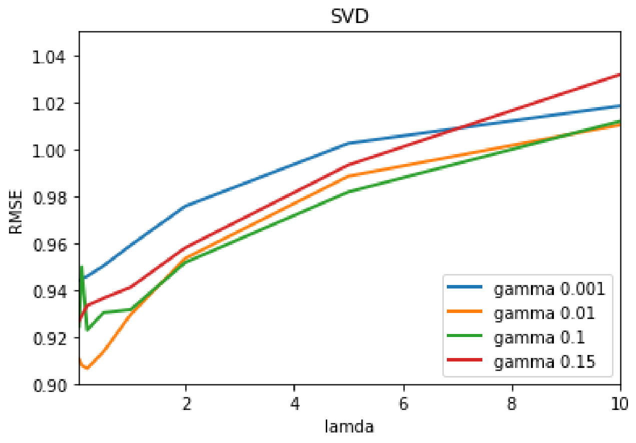
(b)

As the number of neighbors increases the RMSE tends to become constant because we choose only top k similar users based on Pearson correlation. The curve becomes constant when K exceeds the no of users rated that movie



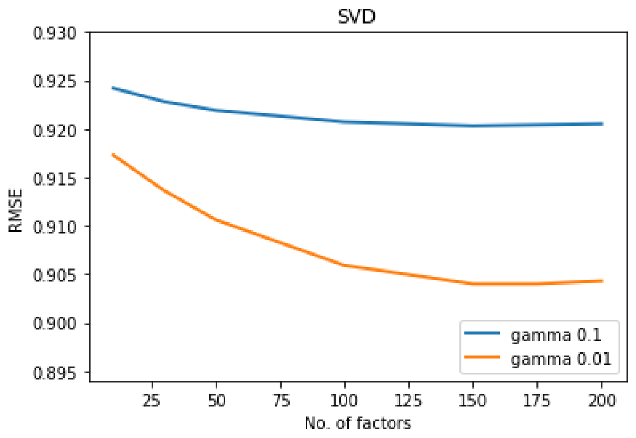
(c)

The number of movies is high, and we can find many correlated movies for a given movie and when K crosses 100 it starts considering the negatively correlated movies that were rated by the user, so it increases the RMSE.



(d)

The above plot shows the variation of lambda. As we increase the lambda the model underfits and results in high RMSE because it controls the user, item bias, user interest and item factor.



(e)

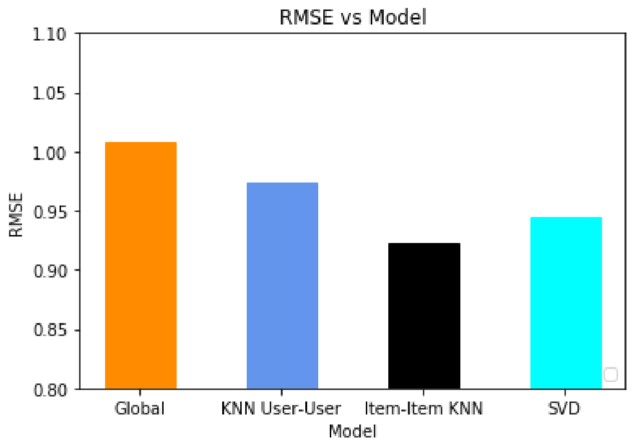
SVD decomposes the utility matrix into low-rank matrices with rank as No of factors. No of factors corresponding to the number of movie genres in the data set. Once the matrix is represented by the rank same as the actual number of movie genres the RMSE becomes constant

4. FUTURE SCOPE:

Restricted Boltzmann machine(RBM) which is an unsupervised learning artificial neural network can be used to predict rating after learning the distribution of the user's in the training phase. Hybrid techniques which use a blend of RBM and SVD can lead to less RMSE.

5. CONCLUSION:

Item to item outperforms all the other methods, however, it is very costly operation compared to SVD because it needs to compute the correlation between all the movies pairs which is O(n^2) where n is no of movies in the dataset. SVD takes less time in decomposing the matrices. If the no of movies is small, then KNN item-item should be preferred for better RMSE otherwise SVD would be a better choice.



REFERENCES:

[1] Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor: Recommender Systems Handbook. 2011.

[2] Phil Chen and Dan Posch: Collaborative Filtering on Very Sparse Graphs A Recommendation System for Yelp.com. 2012. Stanford University

[3] Youchun Ji, Wenxing Hong, Yali Shangguan, Huan Wang*, Jing Ma: Regularized Singular Value Decomposition in News Recommendation System. 2016. ICCSE Japan

[4] Rahul Makhijani, Saleh Samaneh, Megh Mehta: Collaborative Filtering Recommender Systems. Stanford University

[5] Daniel Poon, Yu Wu, David (Qifan) Zhang: Reddit Recommendation System. 2011. Stanford University

[6] Yehuda Koren, Robert Bell and Chris Volinsky: Matrix Factorization Techniques for Recommender Systems. 2009. IEEE computer society.

[7] Prateek Sappadla, Yash Sadhwani, Pranit Arora: Movie Recommender System. New York University.