CS F469, Information Retrieval: Assignment-2

CF based recommender system

Aditya Bodade, Egna Praneeth Gummana, Utkarsh Srivastava, Shivansh Rustagi, and Shubham Agarwal

Birla Institute of Technology and Science, Pilani, Rajasthan-333031, India

Abstract. Collaborative filtering (CF) is a popular technique for rating prediction in recommender systems. CF tries to predict the user's rating on an unseen item based on other similar users' ratings. Computing similarity between users is dominantly carried out using correlation methods such as the Pearson correlation coefficient. These methods compute similarity only based on co-rated items. We would be looking at the implementation of a traditional user based Collaborative Filtering recommender system in Part A, and the problems and limitations faced by this model. Further in Part B, we will delve into modifying the basic CF model to rectify the limitations arising in Part A.

Keywords: Recommender System · Collaborative Filtering .

1 Part A

1.1 Introduction

Every day, millions of new content are published on the Internet. Finding suitable contents is a serious challenge, which all of us face. To meet this challenge, Recommender systems are utilized. They try to recommend a content that is closer to users' preferences. One of the powerful techniques to build a Recommender system is Collaborative Filtering (CF). This technique is currently used in different applications, especially in recommending products such as movies, music, and books. Moreover, in practical applications, several popular e-commerce websites use CF for recommendation, e.g. Amazon and eBay.

The main contributions detailed in this paper are the following. First, designing a user-based Collaborative Filtering recommender system.Second, evaluating our system using dataset consisting of various users, movies, movie tags and the ratings provided by the users to predict the rating for a target user for an unseen movie.Third, a novel approach for improving the recommender system is introduced. Challenges and improvements have been stated.

1.2 Dataset Statistics

The dataset used for the CF recommender system is collected from MovieLens [2], (http://movielens.org), a movie recommendation service. It contains 100836 ratings from 610 users and 3683 tag applications across 9742 movies. Every user has rated at least 20 movies.

1.3 Model Definition

1.3.1 Utility Matrix The rating data that is input to a collaborative filtering system is often referred to as a Utility matrix where each column is associated with an item in i, and each row contains the ratings of the items by an individual user. We have initialised the entries of the Utility Matrix with the corresponding ratings provided by different users and, *NaN* for which there are no provided ratings.Thus,

$UM[i][j]$ = rating for movie j provided by user i,
$UM[i][j]$ = NaN *(if user* i *has not rated movie* j*)*
Utility Matrix has a dimension of |users| ×|movies|.

1.3.2 Neighbourhood selection through Correlation User-based CF technique tries to predict users' ratings on an unseen item based on similar users' ratings. Thus we need to define a correlation between similar user. We have used Pearson Coefficient to compute the likeness of users based on prior ratings.

Similarly let $I_k^{(r)}$ be the set of items rated by the user u_k, and $I_k^{(u)}$ be the set of unrated items by the same user.
Let i be the co-rated items by two users, u_a, u_b i.e. $i \in I_a^{(r)} \cap I_b^{(r)} = I$. $r_{u_k}(i)$ be the rating of item i by the user u_k.
Pearson Coefficient for a user u_a, u_b $\in U$ *(where U is the set of all user)*:

$$sim(u_a, u_b) = \frac{\sum_{i \in I} (r_{u_a}(i) - \bar{r}_{u_a}) \times (r_{u_b}(i) - \bar{r}_{u_b})}{\sqrt{\left(\sum_{i \in I} (r_{u_a}(i) - \bar{r}_{u_a})^2 \times \sqrt{\left(\sum_{i \in I} (r_{u_b}(i) - \bar{r}_{u_b}\right)^2}}}$$

Once the similarity of the active user with all other users has been computed, a method is required to calculate the ratings for each item $i_j \in I_a^u$. We have used the Resnick's prediction formula given below :

$$\hat{r}_{u_a}(i_j) = \bar{r}_{u_a} + \frac{\sum_{u_k \in U} sim(u_a, u_k) \times (r_{u_k}(i_j) - \bar{r}_{u_k})}{\sum_{u_k \in U} sim(u_a, u_k)}$$

1.4 Evaluating CF Recommender System using K-fold cross validation scheme

1.4.1 Cross Validation: Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. We have chosen k as 5 for our evaluation.

1.4.2 Mean Absolute Error: We use Mean Absolute Error (MAE) metrics to measure prediction quality of CF methods. It is one of the most popular measurement methods in CF literature. MAE is defined as:

$$MAE = \frac{\sum_{u_k} |r_{u_k}(i) - \hat{r}_{u_k}(i)|}{N}$$

where $r_{u_k}(i)$ is the rating that user u_k gave to item i.and $\hat{r}_{u_k}(i)$ designates the predicted rating of user u_k gave to item i, and N denotes the number of tested ratings.

2 Part B

This section covers the drawbacks faced by the traditional CF Recommender system and tries to build upon it, rectifying some of the problems

2.1 Challenges faced in CF-RS built in part A

Sparsity: The total number of available ratings, are just a very small fraction of the total number of items available in a database, and hence leads to even very popular items, being rated by only very few of the total number of available users. This problem has a major negative impact on the effectiveness of a collaborative filtering approach. Because of sparsity, it is possible that the similarity between two users cannot be defined, rendering collaborative filtering useless. Even when the evaluation of similarity is possible, it may not be very reliable, because of insufficient information processed. Also, the cold-start problem emphasizes the importance of sparsity.

Cold-start: refers to the situation in which an item cannot be recommended unless it has been rated by a substantial number of users. This problem applies to new and obscure items and is particularly detrimental to users with eclectic taste. Likewise, a new user has to rate a sufficient number of items before the recommendation algorithm be able to provide reliable and accurate recommendations.

Huge Space and Time complexity: The space and iterative time complexity of the CF approach demands O(n) complexity with n being in millions or billions in actual datasets. Also, the approach requires repeated use of the entire dataset, for each similarity and prediction making step, increasing the overheads beyond acceptance.

Scalability: CF algorithm is having a complexity that is already too large. As well, many systems need to react immediately to online requirements and make recommendations for all users regardless of their purchases and ratings history, which demands a higher scalability of a CF system.

Hard to include side features for query/item: The CF vanilla method focuses primarily on predictions based on "wisdom of the crowd" and lays no emphasis on content, demography of other inherent features of the User and/or the item.

Number of out of bound values is high: The vanilla CF approach fails to keep predictions within the acceptable range, and values might lead to misleading recommendations and predictions for ratings.

In nearest-neighbor approaches , the set of sufficiently similar neighbors might be too small to make good predictions and neighbours that might be very distantly similar say, 0.0003, may also be considered in CF nearest-neighbor approaches.

2.2 Proposed Improvements

Innovation 1: [**3**] Suitably adjusting the weight when the number of co-rated items is low. We propose doing this using a linear approach, linearly reducing the weight when the number of co-rated items is low .

Innovation 2: [**1**] We propose giving more weights to "very similar" neighbors, i.e., where the similarity value is close to 1. We propose this by raising the similarity values to a power of 2.5, which leads to weight amplification for users that are very close and vice-versa.

2.3 Addressing issues using these Improvements

2.3.1 Case Amplification [**1**] It tries to find users that are similar to the active user (i.e. the users we want to make predictions for), and uses their preferences to predict ratings for the active user. This is a simple one - we simply amplify each weight by an exponent so that higher weights get higher and lower ones get lower. It tends not to work very well, but you can see a tiny improvement.
We transform the estimated weights as follows:

$$w'_{a,i} = \begin{cases} w_{a,i}^{\rho} & \text{if } w_{a,i} \geq 0 \\ -\left(-w_{a,i}^{\rho}\right) & \text{if } w_{a,i} < 0 \end{cases}$$

The transform emphasizes weights that are closer to one, and punishes low weights. A typical value for ρ for our experiments is 2.5.
The idea of case amplification is to increase the spread of similarity values by raising the similarity to a power of 2.5.

- The quality of predictions are rather good.
- This is a relatively simple algorithm to implement for any situation.
- It is very easy to update the database, since it uses the entire database every time it makes a prediction.

Improvement: Nearest neighbour approach- Since data is very sparse and using nearest neighbour approach we end up considering users as neighbours for the target user even with small similarity values. Using this approach helps us to give more weight to similar users in the neighbourhood as well.

2.3.2 Significance Weighting: [3] Suitably adjusting the weight when the number of co-rated items is low. We propose doing this using a linear approach, linearly reducing the weight when the number of co-rated items is low.
We employed the number of co-rated items for significance computation. This number represents the reliability magnitude of each similarity value. It is complementary of correlation in similarity computation. As the number of co-rated items increases, the higher weight should be assigned. Utilizing this number as weight can improve the CF performance. Accordingly, we modify the similarity measure as:

$$sim'(u, v) = |I_u \cap I_v| \times sim(u, v)$$

where $|I_u \cap I_v|$ is the number of items, which user u and v rated in common. The above similarity measure combines likeness and reliability together. The measure of $|I_u \cap I_v|$ does not require to be normalized owning to scale invariability of CF prediction formula.

Improvement: Number of co-rated items is very less-Number of common items between users represents the reliability magnitude of each similarity value. It is complementary to correlation in similarity computation. As the number of co-rated items increases, the higher weight should be assigned. Utilizing this number as weight can improve the CF performance.
Since the number of co-rated items is very less in our data, it helps us to avoid the cases where less similar users ended up getting high similarity values.

2.4 A corner case (if any) where this improvement might not work or can have an adverse effect.

In spite of the increase in accuracy of predictions, the models fail in covering a few corner cases:

Innovation 1: Weighing similarity based on the number of co-rated items:

- Weighing similarity based on the number of co-rated items: If the number of co-rated items is less than a certain threshold for a lot of items, then the similarity values become misleading, and predictions might get worse.
- Since multiplying the similarities with the number of co-rated items makes the resulting similarity of our approach increase beyond than 1, it causes difficulty to interpret the similarity value.

Innovation 2: Amplification of similarity values:

- If there are no neighboring users having a similarity value that is close to 1 for a user(s), then the model might give less weight to the peer ratings over the target item.
- When the co-rated items are less, high similarities are resulted from this approach. Thereby giving high weightage to a user, even though the user are not actually similar mutually.

2.5 Demonstrate the actual impact of the improvement. Give three examples, where the improvement yields better results compared to the part A implementation.

Innovation 1: Weighing similarity based on the number of co-rated items:

- Suppose there is one co-rated item. (which happened in our case) and it resulted in a similar user, which is actually an incorrect prediction as we have only one peer for calculation of similarity and the prediction value. In such cases, the innovation yields more accurate results.

- In cases such as users having high correlation, the similarity has been weighted irrespective of the number of co-rated items, but the expectation is that high correlations should have a higher influence in decreasing prediction error. This issue has been addressed in the work.

- There are cases where the user-item rating vectors vary in the number of co-ratings, and the intersections play an important role in deciding the similarity of users. Unlike the normal approach, our work helps to linearly weigh the similarities based on the number of co-rated items, thereby reducing error in predictions.

Innovation 2: Amplification of similarity values:

- We are giving high weightage in the Resnick prediction formula to the highly similar users. This increases the likeliness that the target user will like the item. (For eg. high weightage is given to the movies seen by my friends who have a similar interest).

- The users who have very low similarity with the target user, get a significantly lower weightage, as compared to the normal approach due to the exponential amplification, thus having very little impact on the final predictions for the target.

- The similarity weights considered for user-rating predictions are secluded into farther distant clusters, as a result of the exponential powering. In cases, where the mix is heterogeneous, this helps to further decrease prediction error.

2.6 What is the significance of multiplying the value of rv,m by the similarity of user u to user v, in the Resnick prediction formula?

Resnick's prediction formula weighs the contribution of a partner's prediction according to its **degree of similarity** with the target user so that more similar partners have a large impact on the final ratings prediction. The notion to be satisfied while making predictions for user-items is to weigh the rating **higher** if someone present in the neighborhood of the user, **agreed more** with the user in the past and vice versa. We are using the rating of similar users, therefore we need to use the rating in the order of similarity values.

2.7 Exploiting additional possible information from data to improve the existing system developed in part A:

Possibly, some additional statistical trends can be exploited from the information in the dataset to improve the prediction accuracy and model efficiency.

- Commonly liked items are not so informative as agreement on controversial items, so possibly analyzing the variance in the ratings of items and weighing items that have a higher variance, could be a possible improvement.
- On similar lines, rarely rated items if common to users indicate a higher level of similarity between users, and tracking the number of frequently rated items as compared to the rarely rated ones and exploiting the related statistics in the dataset, might yield a better selection of neighborhood, improving the RS model.
- The transitive properties between users could have been exploited by using recursive, or by graphical (1,5) links traversal approach, and could have helped yield an improved outcome, by incorporating better Nearest Neighbourhood, and hence, more accurate predictions.

3 Notes on Implementation and Assumptions

3.1 Efficient implementation:

1. Using NumPy instead of pandas for the required user-item matrix: NumPy is faster than pandas.

2. Predict Ratings Function has been optimized for use, by predicting over Only required sets of movies, eliminating the overhead of predicting all 9742 movies all the time.

3. MovieIds have been mapped to their index values, to reduce the range, and hence the spanned space of column size, from 193609 to 9742, reducing computation time drastically.

4. Vectorized implementation through NumPy instead of iterative, hence decreasing the computation time.

5. Auto Broadcasting, of Matrices and Vectors reduce operational overheads and are able to handle computations very efficiently.

6. All the filterings and preprocessing operations have been implemented in the most efficient approach through the use of Numpy vectors.

7. Using advanced algorithms and utility functions form sklearn library, that allow code modularity, abstraction and convenient performance optimisation.

3.2 Assumptions:

1. Preprocessing assumption: Items which have been rated by less than or equal to 5 users have been removed from the dataset.

2. The outputs yield from the prediction function, are truncated within the limits, to bound the ratings to the valid bounds [1,5].

3. All "Unable to Predict" (due to lack of ratings, Cold Start issues) are considered as NaNs.

4. All NaNs have been disregarded in the computations of Mean Absolute Error, for the sake of veracity.

5. Only the intersected set of available ratings have been used for similarity calculation between any two users.

6. Filtering has been carried out before considering the top-k, to remove NaNs from the ratings to be considered for prediction.

3.3 Result

MAE scores corresponding to different K folds

K	Vanilla approach	Significance weighing	Amplification
1	0.42013	0.38758	0.30862
2	0.42351	0.38935	0.31309
3	0.43060	0.39936	0.32018
4	g 0.42209	0.39479	0.30987
5	0.41946	0.38970	0.31378

4 Conclusion

In this report, we present our assignment on User Based Collaborative Filtering Recommender System. We have used the Movie Lens dataset for this project. We have used the vanilla approach for the partA of the assignment as required. We have used the Pearson correlation coefficient and the Resnick prediction formula. We proposed two innovations to improve the recommendation accuracy. They are Case Amplification and Significance weighting. We have implemented the same and presented our results using the k-fold cross validation (k = 5). The evaluation metric shows the steady improvement in result, that was expected according to the distribution of dataset.

References

1. John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.
2. F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.
3. Mohsen Raeesi and Mehdi Shajari. An enhanced significance weighting approach for collaborative filtering. In *6th International Symposium on Telecommunications (IST)*, pages 1165–1169. IEEE, 2012.