How do I get a good movie recommendation from Netflix – Collaborative Filtering

Jaehyun Yoon
University of Maryland,
College Park
jyoon125@umd.edu

***Abstract*— As we were briefly introduced to a recommendation problem as an example for one of the optimization problems during the class, I thought that this project would be a great opportunity to find out how I get a good recommendation when I watch YouTube and Netflix and how can we solve optimization methods using methods that we were throughout the course. This project is focusing on implementing two collaborative filtering methods as Item-based Collaborative Filtering (CF) and Latent factor to get a movie recommendation.**

Keywords— Recommendation algorithm, Optimization, Collaborative Filtering, User-based CF, Item-based CF, Latent factor, SVD, Cosine similarity

I. INTRODUCTION & MOTIVATION

Probably, most people, whether they recognize it or not, have experienced at least one recommendation algorithm in our life. For example, when you visit Amazon for shopping, you can see that the item is recommended to visitors, or when you click to get information on a specific product, you can select a product in various ways, such as recommended items, popular items, and items you may like. Another recommendation example is that you would ask for a review to your friends about the new movie if they already watched the movie before.

According to an interview with The New York Times, Neal Mohan, YouTube's Chief Product Officer, said, '70% of YouTube users' viewing time is the result of the recommendation algorithm, and the introduction of the algorithm increases the total video viewing time by more than 20 times than before [1]. Netflix also revealed through self-assessment that 75% of sales are generated by the referral system [2]. In other words, these two video platforms have strong algorithms to recommend perfect videos to users and personalized content. It can be said that they are guarding the throne even at present when the online video platform opponents appear one after another.

Collaborative Filtering (CF) is a technique of recommending a product or contents when other users similar to a specific user give good evaluations or if there is a good evaluation on other products similar to a specific product. There are two types of collaborative filtering: Nearest neighbor and Latent factor. There are many Collaborative Filtering (CF) research, methods, and developments that provide recommendation systems to. I found and used resources that are similar to this topic, especially, Abhinav Ajitsaria's "Real Python" blog, "Build a Recommendation Engine With Collaborative Filtering" [3] and one of class from Carnegie Mellon University, "Matrix Factorization and Collaborative Filtering", taught by Matt Gormley [4], which is the resources I refer to the most for this optimization problem.

In this project, I explore one of the types of Nearest neighbor methods, Item-based CF, but mainly focus on solving Latent factor, which is about using Singular value decomposition (SVD) and Gradient descent method to optimize the cost function so that the predicted vector R-value based on the vectors P and Q have the least error with the actual R-value. The dataset contains the 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users from the education and development version of MovieLens at Grouplens.org [5].

II. METHOD

Collaborative filtering (CF) is a method of grouping people and recommending contents or products that people in the group commonly viewed or purchased. The basic idea of CF is "Isn't it very likely that I will like the content selected by people like me?" The main purpose of CF is to predict the rating of items that the user has not yet rated. As I mentioned in the introduction, there are two types of Collaborative Filtering: Nearest neighbor and Latent factor algorithm, and Nearest neighbor has two methods which are User-based CF and Item-based CF.

A. Nearest neighbor theory

1. Cosine similarity

The cosine similarity refers to the degree of similarity between two vectors that can be obtained using the cosine angle using the dot product between two vectors and divided by the norm of each vector. As the two vectors are in the same direction, the cos similarity becomes larger. If the two vectors are in the same direction, the result is 1, and in the opposite case, the cos value is -1.

$$cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}||\vec{B}|}$$

2. User-based CF

In User-based CF, the degree of similarity is based on how many similar items are preferred by two users. Users who have purchased similar products with a specific user are considered similar users and recommend other products that similar users prefer based on users who are similar to a specific user. The procedure to get User-based CF is similar to Item-based CF, but

the method uses the dataset that has the user information as a column and item information as a row.

3. Item-based CF

In general, Item-based CF is preferred over User-based CF. This is because it is difficult to judge that they are similar people in many cases just because they bought the same product. The method uses the dataset that has item information as a column and user information as a row. Products with similar ratings to specific products from users are considered similar products.

a. Weighted Rating sum

The weights should be applied by assigning ratings according to similarity. The similarity and rating data of each user are multiplied each other, and then divided by norm to normalize like the following formula. The Items with higher similarity are multiplied by two values to give a higher rating [3].

$$\hat{R}_{u,i} = \frac{\sum N\left(S_{i,N} * R_{u,N}\right)}{\sum N\left(|S_{i,N}|\right)}$$

$\hat{R}_{u,i}$: Personalized predicted score value for user u and item i
$S_{i,N}$: Similarity vector of Top-N items with the highest similarity to item i
$R_{u,N}$: Actual rating vector for the Top-N items with the highest similarity to item i of user u

b. The order of implementation

1.Convert user-item matrix data to item-user matrix data.
2.Calculate item similarity by cosine similarity between items.
3.Calculation of predicted score reflecting the similarity between items among items not viewed by the user.
4.Recommend items in order of highest predicted score.

B. Latent factor Latent factor

The latent factor is a method of extracting the rating by the result of dot product from decomposed matrices using Singular value decomposition (SVD) and Gradient descent [3].

1. Singular Value Decomposition (SVD)

$$SVD = U\Sigma V^T$$

Based on the original data set of the reconstructed matrix and the rating data set in another dense form, the rating can be made based on predictable latent factors while reconstructing the predicted ratings for items not selected by the user. Latent factor collaborative filtering is a technique that enables recommendation prediction by extracting potential factors hidden in the user-item rating matrix. We will solve this problem by assuming that the genre of the movie is a latent factor [3][6][7]. I recompose the vector R using SVD as following [6].

$$R \underset{by}{\overset{Decomposed}{\Longrightarrow}} P * Q^T$$

P is the m * k dimension user-latent factor matrix with the value of the relationship between the user and the latent factor.

- Each row reflects latent factors for individual users.

Q is an item-latent factor matrix of n * k dimension with values of the relationship between the item and the latent factor.

- Individual columns reflect latent factors for individual item.

2. Gradient Descent method

The SVD can be applied to matrices without missing values. Therefore, we cannot decompose the P and Q matrices in the usual SVD way. Using gradient descent, P and Q are inferred through cost function optimization so that the predicted R values based on P and Q have the least error with the actual R-value.

a. Optimization

The objective function is we minimize the errors between actual r and predicted values p and q. This is also known as Stochastic Gradient Descent (SGD) that minimizes the total distance to the sample. In the process of minimizing errors, we apply normalization to prevent overfitting as the following equation [8].

$$\min_{p*,q*} \sum \left(r_{(u,i)} - P_u * q_i^t\right)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

SGD's equation can be called a non-convex problem because two unknown vectors p and q are related, but if one of the unknowns is fixed as a constant, this optimization problem becomes a quadratic equation to find an optimized solution. Therefore, the remaining functional vectors are fixed as constants until the optimal solution is obtained. In other words, in order to calculate the user feature vector, you can first think of a method of fixing the item feature vector as a constant and solving it with the least squares. As we update our values p and q, we look for the minimal value at each point [6].

$$\acute{p_u} = p_u + \eta(e_{(u,i)} * q_i - \lambda * p_u)$$
$$\acute{q_i} = q_i + \eta(e_{(u,i)} * p_i - \lambda * q_i)$$

p_u : the user u row vector of matrix P
q_i^t : the item i column vector of matrix Q
$e_{(u.i)}$: $r_{(u,i)} - \hat{r}_{(u,i)}$: Difference error between the actual and predicted matrix values located in row u and column I
λ : lambda value

b. The order of implementation

1. Set P and Q as arbitrary-valued matrices.
    a. In our case, P and Q are filled with zeros and then compared with the actual values.
2. Compute the predicted R matrix by multiplying the P and Q^T values and compute the error values corresponding to the predicted R matrix and the actual matrix.
3. Update the P and Q matrices individually with appropriate values to minimize this error value.
    a. We need to get the partial derivative and update it to the value based on the derivative.
4. Recommend items in order of highest predicted score.

III. IMPLEMENTATION

A. Nearest neighbor – "Item-based CF"

a. Data pre-processing (pivot table and transpose)

I merge two tables ratings and movies using a primary key which is movieId and create a data frame with movieId as the column, userId as the row, and rating as the value of a table using pivot table Since we are conducting Item-based CF, I have to switch the column and row which is similar to transpose of the vector.

b. Cosine similarity

The sklearn python library package has been making our life easier when we need the cosine similarity. I convert the NumPy matrix returned by cosine_similarity() to DataFrame by mapping the movie name.

c. Weighted Rating sum

The similarity and rating data of each user u are conducted by the dot product of each matrix and then divided by norm to normalize The vector S value has the shape of (9719, 610) and the R-value has the shape of (9719, 9719). I calculate the Mean square Error (MSE) for predictive performance evaluation after weighting rating. I refer to one of the GitHub link to find the Top-20 similarity. I Initialize a prediction matrix filled with zeros as large as the user-item rating matrix and loop through the column size of the matrix to compute the indexes of n data matrices in the order of similarity in the similarity matrix [3][9].

d. Result

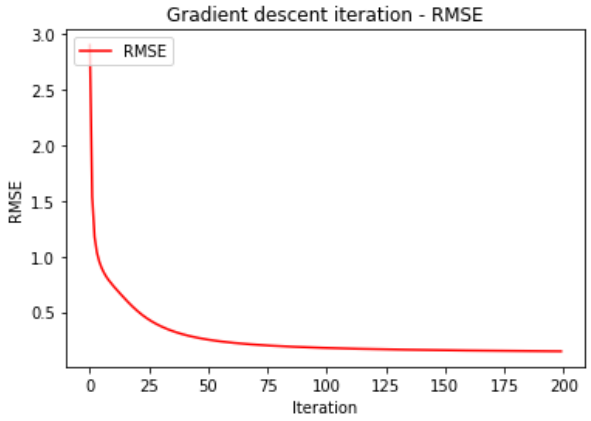
The first Mean Square Error (MSE) which has any constraint or limitation after the weighting sum is way higher than the top-20 similarity. In other words, the constraints are to calculate the similarity between the target user u and all other users, select the top n similar users, and take a weighted average of ratings from those n users who weigh the similarity [6]. The result of Item-based CF provides the movie recommendation through item-based CF among movies that users have not watched. I pick a random user which is the number 7 to see what are the recommended movies from Item-based CF. The result is following :

title	pred_score
Finding Nemo (2003)	2.955692
Bourne Identity, The (2002)	2.921060
Matrix, The (1999)	2.328329
Cast Away (2000)	2.171403
Italian Job, The (2003)	2.171037
Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	2.052555
Green Mile, The (1999)	2.010310
Pirates of the Caribbean: Dead Man's Chest (2006)	1.950853
Shawshank Redemption, The (1994)	1.942261
Titanic (1997)	1.929389

B. Latent factor

a. Singular Value Decomposition (SVD) and Stochastic Gradient Descent (SGD)

I specify the size of the P and Q matrices and initialize the random values with a normal distribution, and then save the row position, column position, and value with R> 0 in a list object. AS I update P and Q using Stochastic Gradient Descent (SGD) with the alpha values as 0.01 and the lambda value as 0.01, I look for the error value that is the difference between the actual value and the predicted value. This will take a few minutes to compute the value. I also plot how the RMSE value is changing for iteration, which I set it up for 200. The RMSE value is dropped from 2.9 to 0.7 after the first iteration as following.



b. Result

The result of Item-based CF provides the movie recommendation through item-based CF among movies that users have not watched. I pick a random user which is the number 7 to see what are the recommended movies from Item-based CF. The result is following :

title	pred_score
Twister (1996)	5.537622
Schindler's List (1993)	5.478221
Ghostbusters (a.k.a. Ghost Busters) (1984)	5.459786
Snatch (2000)	5.346037
Speed (1994)	5.285207
Christmas Story, A (1983)	5.262388
Pulp Fiction (1994)	5.179681
Reservoir Dogs (1992)	5.144662
Men in Black (a.k.a. MIB) (1997)	5.132820
Go (1999)	5.114149

IV. CONCLUSION

I now know the Collaborative Filtering system, User-based CF, Item-based CF, and Latent factor methods which are one of methods in recommending system. I'm sure this is relatively simple, and the real-life problems that Netflix or YouTube use way more complex than this problem, but I get my foot into it and learn a new thing about the system. The predicted score shows that the Latent factor would give a better recommendation based on these methods, but I'm not able to check if those are. I'm surprised that two different methods give me different results which there is not a single common movie between the two methods. This project also helps me to understand better the gradient descent method especially the Stochastic Gradient Descent (SGD) method, and the Singular Vector Decomposition (SVD) is one of the ways that people use to win Netflix recommendation competition [8]. Last but not least, I learn how optimization problems can prevent overfitting the data and how important they are to solve machine learning problems.

REFERENCES

[1] Roose, K. (2019, March 29). YouTube's product chief on ONLINE radicalization and Algorithmic rabbit holes. Retrieved February 19, 2021, from https://www.nytimes.com/2019/03/29/technology/youtube-online-extremism.html

[2] Roose, K. (2019, March 29). YouTube's product chief on ONLINE radicalization and Algorithmic rabbit holes. Retrieved February 19, 2021, from https://www.nytimes.com/2019/03/29/technology/youtube-online-extremism.html

[3] Ajitsaria, A. (2021, February 13). Build a recommendation engine with collaborative filtering - Real Python. Retrieved February 19, 2021, from https://realpython.com/build-recommendation-engine-collaborative-filtering/

[4] Gormley, M. (2017, April 19). Matrix Factorization and Collaborative Filtering.Retrieved201,from https://www.cs.cmu.edu/~mgormley/courses/10601-s17/slides/lecture25-mf.pdf

[5] Grouplens.org, MovieLens Latest Datasets https://grouplens.org/datasets/movielens/

[6] Luo, S. (2019, February 06). Intro to Recommender SYSTEM: Collaborative filtering. Retrieved February 19, 2021, from https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

[7] Paterek, A. (n.d.). Improving regularized singular value decomposition for collaborative filtering. Retrieved February 18, 2021, from https://www.mimuw.edu.pl/~paterek/ap_kdd.pdf

[8] Gower, S. (2014, April 18). Netflix Prize and SVD. Retrieved February 18, 2021, from http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf

[9] https://github.com/AshwathSalimath/Movie-ratings-prediction-using-MovieLens-Dataset/blob/master/matrix_factorization.py