Movie Recommendation System Considering Bias in Ratings

Jingci Wang
Khoury College of Computer Sciences
Northeastern University
Boston, MA
wang.jingc@husky.neu.edu

Junmei Luo
Khoury College of Computer Sciences
Northeastern University
Boston, MA
luo.jun@husky.neu.edu

ABSTRACT

With massive data emerging on the Internet, people are troubled with finding effective information. Recommendation system can provide potential and meaningful recommendations to users, especially for movies and TV shows on the website. In this project, we aim at building movie recommendation systems. Different methods popularly used in recommendation systems are implemented, including popularity-based model, user-based, item-based and hybrid collaborative filtering model, and matrix factorization methods such as SVD and latent factor models. Meanwhile, we consider the effect of bias terms and see if we can improve the models. Finally, root-mean-square error (RMSE) is used to evaluate and compare the models.

KEYWORDS

Recommendation System, Collaborative Filtering, Matrix Factorization

1 INTRODUCTION

This project is to build a movie recommendation system. Recommendation systems are built to recommend new movies to users, to dig out what users may like but they did not know before. Nowadays, with the rapid development of science and technology, movies and TV shows are popularly watched on the Internet, which allows us to collect data from every individual user and can do personal recommendations.

Recommender systems are widely used in movies and TV shows services such as Netflix and YouTube. Finding the potential movies that users may like, recommender systems aim to save energy and time for both users and movie websites, as well as to decrease the possibility of being frustrated when searching for movies and developing the film market.

The key problem for our project is to predict the rating for a movie that the user has never rated before. Based on previous behaviours, it predicts the likelihood that a user would prefer an item and what is more, the rating this user might give to the item. For example, Netflix uses recommendation system. It suggests people new movies according to their past activities that are like watching and voting movies.

In the meanwhile, there can be some bias in movie ratings. To be more specifically, some movies are universally loved or hated but some users may have niche interests, so that the ratings cannot reasonably represent peopleâĂŹs opinions of movies; some users are pickier than others which will result in high or low scores for all the movies rated by one person regardless of preferences because of the lack of uniform standards for ratings. To tackle these problems, we plan to do some comparative study on the effects of bias terms.

The primary goal of this project is to predict the rating given a user and a movie, mainly using three different methods - popularity-based recommender system, collaborative filtering and matrix factorization models on a subset of MovieLens 20M dataset[10]. We describe the results using each of the models and compare them in the scenario whether or not bias terms are included.

We would use rooted mean square error (RMSE) as the metric to evaluate our model performance.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{xi}(r_{xi} - \hat{r}_{xi})^2} \tag{1}$$

2 RELATED WORK

In recent years, recommendation systems play a more and more important role in the business development. By reducing the possibility of finding a less interested movie because of information overload, the system saves time and energy for both users and companies. Thus, there are more and more researches on recommendation systems. Content based, collaborative filtering and hybrid approaches are commonly used in recommender systems.

Our project works on the study of Mining of Massive Datasets [9], and with limitation to the dataset, we cannot do content-based model. However, we can try all other popular approaches. Cold start is a major challenge for recommendation system, because recommendations are given based on history records, but new users may not have histories. To deal with this, many researchers use content-based and ask users to submit their interests such as tags and keywords[8]. For our problem, we can make use of the ratings, so we build popularity-based model.

The most commonly used algorithm in recommendation systems is Collaborative Filtering (CF) [1]. There is user-based CF and item-based CF, and some researchers also proposed hybrid model combining user and item CF together [3].

Matrix Factorization (MF) may perform better from some researchers. One of the well-know algorithm called singular value decomposition (SVD) [2], it is to reduce the dimension and capture the most important factors to do prediction. Furthermore, latent factor model [4] is another one. It has a loss function after performing dimensional reduction, and it can be optimized by using gradient descent. Also, alternating least square can be applied for the recommendation system.

In conclusion, most of the researches focus on single approach. Learning from all of the previous researches, we decide to try both collaborative filtering and matrix factorization methods and to see which one is better. We want to build a comprehensive hybrid model out of user-based and item-based collaborative filtering models which can best make use of the advantages of similarities both between users and items.

3 PROPOSED APPROACH

3.1 Popularity-Based Recommender System

Popularity-Based Recommender System recommends the same movies to every user, and it is mainly to deal with the cold start or nearly cold start problem.

In this case, we would find the top 50 movies with most rating records and recommend them to any user regardless of any preference shown in their profiles.

3.2 Collaborative Filtering

Collaborative Filtering is making recommend according to combination of user's experience and experiences of other users. The rationale behind collaborative filtering is to predict an known rating for a pair of item to user according to the set of similar items or that of users.

3.2.1 *User-Based Collaborative Filtering.*

User-based collaborative filtering is to make recommendations according to combination of user's experience and experiences of other users. First we need to make a user-item matrix. Each row stands for the rating records for a user and each column for an item, in this case the movie Id. Secondly, we compute the cosine similarities between users. Each row is a vector representing a user and we compute similarities of these rows (users). Thirdly, find top k users who are similar to the targeted user based on past behaviours by choosing the ones with highest similarity scores to the targeted user. Finally, we calculate the predicted rating for the target user by calculating a weighted average of the ratings of most top k similar users.

3.2.2 *Item-Based Collaborative Filtering.*

In item-based collaborative filtering, instead of finding relationships between users, it mainly calculates the similarity between items. For any of the users, habits and tastes of users can be changed. This situation makes it harder to recommend. However, attributes of movies do not change, which makes it easier to predict the rating precisely.

The process of building item-based collaborative filtering model is similar to that of the user-based collaborative filtering. At this time, we make an item-user matrix that is the transposed matrix of that in user-based recommender systems. Each row stands for an item (i.e., movie) and each column for an user. The values in the cells are ratings for the corresponding movie-user pairs.

For both user-based and item-based collaborative filtering, our baselines are to use the raw data and directly imputes the missing values with 0. Then, taking item-based collaborative filtering as an example, the predicted value for a given user-movie pair is calculated as follows

$$r_{xi} = \frac{\sum_{j \in N(x,i)} s_{ij} r_{xi}}{\sum_{j \in N(x,i)} s_{ij}} \tag{2}$$

where $N(x, i)$ denotes the set of items most similar to item i, and s_{ij} stands for the cosine similarity between item i and item j.

After that, we make some adjustments on the experiment setting. First of all, instead of using the raw data, we substract the row mean for entries that have a rating which makes our metric equivalent to computing Pearson correlation coefficient. Secondly, borrowing

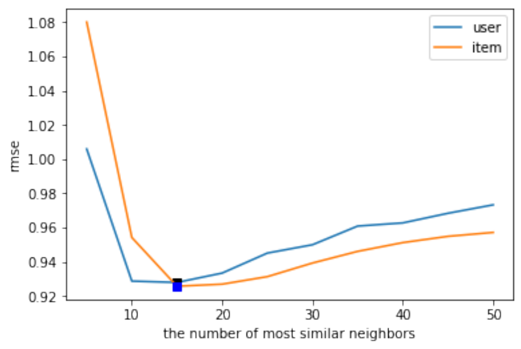

Figure 1: Grid search finding best k for user-based and item-based model

the idea from [9], we also consider the bias terms in user level, item level as well as the overall level. Finally, our objective becomes

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(x,i)} s_{ij}(r_{xi} - b_{xj})}{\sum_{j \in N(x,i)} s_{ij}} \quad \text{where } b_{xi} = \mu + b_x + b_i \tag{3}$$

Here μ stands for the mean overall rating, and b_x denotes the mean rating that user x has given minus mean overall rating and b_i denotes the mean rating that item i has received minus mean overall rating.

To find the best number of neighbours (i.e., the number of most similar items or users for targeted pair) k, we perform a grid search and find the k that give the minimum error. We try k from 5 to 50 with interval 5, and find the best k at 15 for both item-based and user-based collaborative filterings as in Figure 1.

3.2.3 *Hybrid Model.*

After building the user-based collaborative filtering and item-based collaborative filtering, we plan to build a hybrid model out of them. The two collaborative filtering models both have their focal point or emphasis, By combining them, we make best use of their advantages and avoid some disadvantages, to build a more comprehensive model.

The basic idea here is to use a linear combination of the results from the above two collaborative filtering models. Our target is to find a reasonable ratio of one outcome to the other. Thus, the objective becomes

$$Rating_{Hybrid} = (1-\lambda) \times Result_{user-based} + \lambda \times Result_{item-based} \tag{4}$$

To get the ratio that can best predict the rating, we also perform a grid search w.r.t λ from 0 to 1 with interval 0.1, and choose the λ that give the minimum RMSE. The best λ equals to 0.6, as presented in Figure 2.

3.3 Matrix Factorization

3.3.1 *SVD.*

We use SVD as our baseline model for matrix factorization. Firstly we perform SVD on the user-item matrix imputing missing values with 0. Then, we choose the number of reserved ranks (the number

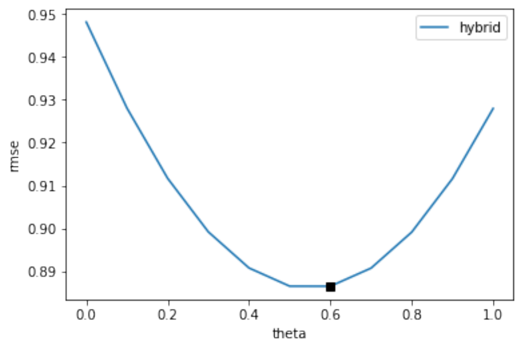

Figure 2: Grid search finding best λ for hybrid model

of singular values) by grid search. The reconstructed rating matrix records the prediction on each entry.

$$R = U\Sigma V^\top \tag{5}$$

where R is the rating matrix.

3.3.2 Latent Factor Model.

In this part, similar to SVD, our goal is to find $R = QP^\top$ that can make the best prediction. In case of overfitting, we also add regularization terms to prevent the lengths of rows in Q and P being too large.

$$\min \sum_{(x,i)\in R} (r_{xi} - q_i p_x)^2 + \lambda_1 \sum_i ||q_i||^2 + \lambda_2 \sum_x ||p_x||^2 \tag{6}$$

As SVD gives minimum reconstruction error, we use $Q = U$ and $P^\top = \Sigma V^\top$ as initialization.

On the other hand, we also include bias term as what we do in collaborative filtering experiment.

$$\min \sum_{(x,i)\in R} (r_{xi} - (\mu + b_x + b_i + q_i p_x))^2 + \lambda_1 \sum_i ||q_i||^2 + \lambda_2 \sum_x ||p_x||^2 \tag{7}$$

Then, with the assistant of stochastic gradient descent, we minimize the cost functions (6) and (7) and obtain the resonable PQ decomposition.

Still, the regularizers are found by grid search.

3.3.3 Alternating Least Square.

With the help of Surprise package [7] in Python library, we also build a recommender system with the strong alternating least square model as our state-of-art algorithm.

4 EXPERIMENTS

4.1 Datasets

The MovieLens dataset[10] is collected and made available from the MovieLens website by GroupLens Research team[5]. The data set contains data from users who joined MovieLens in the year 2000. This is a dataset of about 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags.

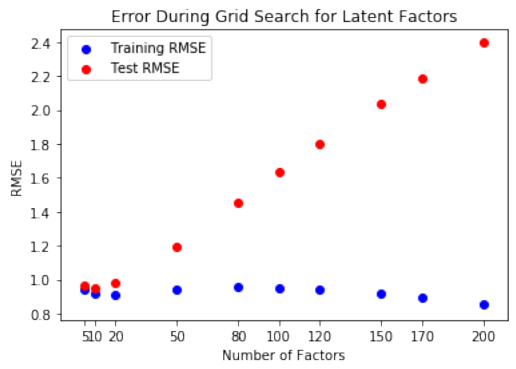

Figure 3: Error during grid search for latent factors (5-200)

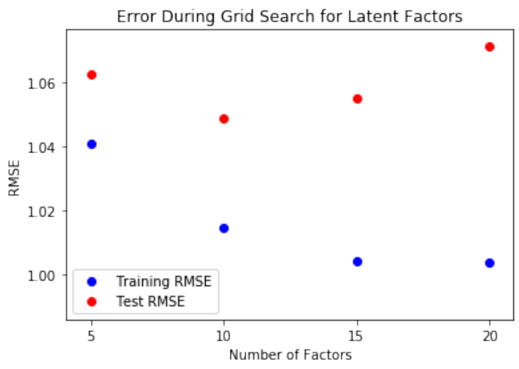

Figure 4: Error during grid search for latent factors (5-20)

One of our main algorithms (collaborative filtering) suffers when encountering users or moives with few or none historical records. Due to the sparsity of the dataset, (99.46% after being transferred into a unity matrix), since we do not have enough information for a content-based recommender which is said to be very useful in dealing with cold start problems, we decide to eliminate the movies that hold less than 1000 recorded ratings and users that have given less 1000 ratings, resulting in records for 553 users and 1884 movies, and sparsity level becoming 30.73%. This action in some way makes the ratings more objective to the taste of most of the public.

Ratings are integers on a 5-star scale. Each user and each movie is identified by a unique id. Figure 5 counts the number of each rating users gave to the movies which indicates that most users tend to give 3 - 4 scores for the movies.

The records are stored in a dataframe in which each row stands for a user-movie-rating pair. To accelerate the computation speed, we perform a 10-fold cross validation in Kaggle kernel for each of the above model.

4.2 Results

As we can see from Table 1, adding bias term significantly improve the performance of collaborative filtering. The combination of item-based and user-based collaborative filtering unsurprisingly gives the better score close to our state-of-art algorithm.

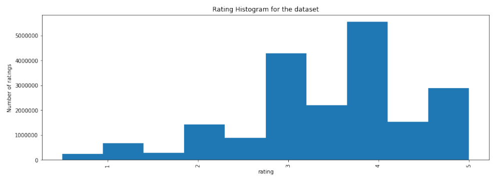

Figure 5: Frequency of each ratings appeared in the data

	Model	RMSE
CF	User-based CF	1.5044
	User-based CF + Bias terms	0.9279
	Item-based CF	1.3257
	Item-based CF + Bias terms	0.9257
	Hybrid (0.4*User + 0.6*Item)	**0.8865**
MF	SVD	1.7200
	Latent Factor Model with SGD	**0.8012**
	Latent Factor Model with SGD + Bias	0.9266
	Alternating Least Square	**0.8170**

Table 1: RMSE results for each model

In term of matrix factorization models, Stochastic gradient descent also greatly reduce the test set RMSE of SVD, even exceeding our state-of-art model (see Figure 6). However, including bias terms in this case fail to outperform the one without bias terms (see Figure 7).

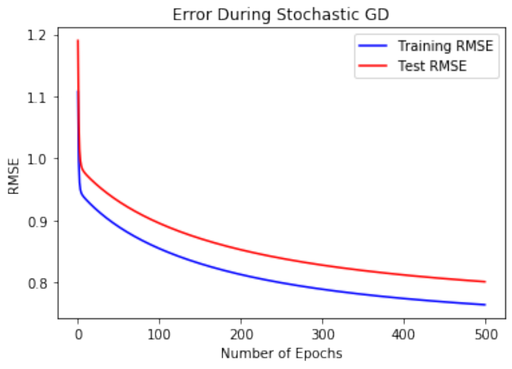

Figure 6: SGD for latent factor model without bias term

Model	Recall	Precision
Popularity-based	0.3479	0.1876
Latent Factor Model	**0.8227**	**0.9133**
Latent Factor Model + Bias	0.7557	0.7354

Table 2: Recall and Precision result on 3 models

Furthermore, we recommend the top 50 most popular movies to 1000 users who rated less than 50 movies. Both recall and precision rate suffers as shown in Table 2, indicating cold start or nearly cold start problem cannot be properly addressed with simply popularity-based model.

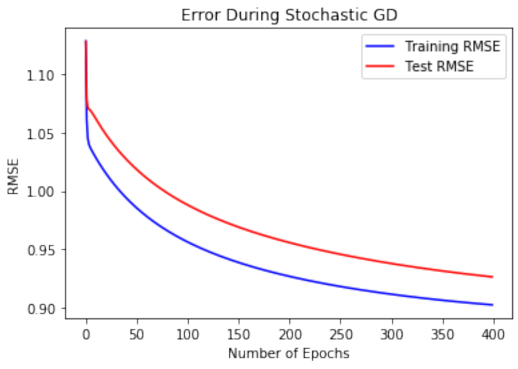

Figure 7: SGD for latent factor model with bias term

On the other hand, we also compute the recall and precision rate for latent factor models, setting the cells with rating larger than 3 to be indications that the corresponding users like the movies correspondingly. The one excluding bias terms still outperform the other one.

5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

- First of all, we compared performances of different recommendation system models and the effects of adding bias terms.
- Secondly, we tuned the hyperparameters in each model by grid search and cross validation achieving testset RMSE close to state-of-art algorithm
- In our experiment, hybrid collaborative filtering model and latent factor model without bias term gave the best results

5.2 Limitations

There are still some issues in our project.

- Firstly, including bias terms fails to give improvement on latent factor model, which is weird. It is likely that we only performed stochastic gradient descent w.r.t Q and P and setting bias terms as parameters to perform gradient descent with rather than simply hypeparameters would give a reasonable result.
- The other things is that collaborative filtering models suffer in terms of computation power. It should be improved with more decent code ethic.

5.3 Future Work

In the following stage, we make several plans for the improvement of our project.

To begin with, since none of our models can properly deal with cold start problem, we plan to extract user and movie information from other resources such as IMDB. Following that, we would be able to build up user-profile and item-profile respectively based on the attributes of each user, reviews or genres of movies and etc. Then, a content-based recommendation system would be ready to deal with cold start or nearly cold start problem.

The next thing is to dig more about the bias term. For instance, in Figure 8 and Figure 9, we visualize the number of movies rated in each year and the average ratings for movies released in each year. The results are quite counter intuitive. It is possible that a year with more movies released tend to give products with low quality. It is possible that nowadays media companies are mastery at promoting newly-released movies but with time going on, people find that the movies are not as excellent as what they were told resulting in the decrease of overall rating. It is also possible that an old movie still replayed by contemporary people is highly likely to be a classic one and audience would like to give it high score and show tolerance and understanding to the technique parts.

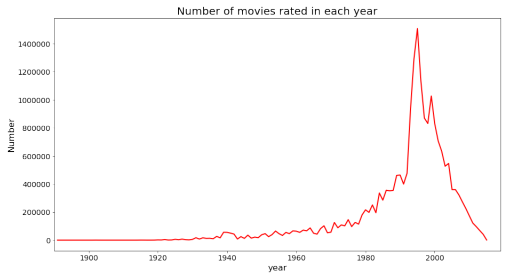

Figure 8: Number of movies rated in each year

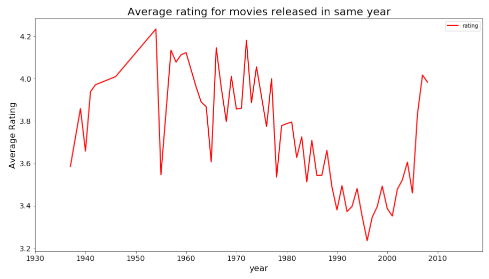

Figure 9: Average rating for movies released in same year

In our future work, inspired by [6] we would also give attempts on some temporal and spatial analysis w.r.t the bias in our model. For example, in the user deviation, use $b_i(t) = b_i + b_{i,Bin(t)}$ rather than simply b_i. Hopefully, this will boost the performance of our methods.

REFERENCES

[1] Billsus D and Pazzani M J. 1998. Learning Collaborative Information Filters. *Icml* 98 (1998), 46–54.

[2] BarragÃąns-MartÃŋnez et al. 2010. A Hybrid Content-based and Item-based Collaborative Filtering Approach to Recommend TV Programs Enhanced with Singular Value Decomposition. Co-Autoregressive Models. *Information Sciences* 180.22 (2010), 4290–311.

[3] Chao Du et al. 2016. Collaborative Filtering with User-Item Co-Autoregressive Models. *Web* (2016).

[4] Yin et al. 2014. Digital TV Program Recommendation System Based on Latent Factor Model. *Applied Mechanics and Materials 513.Applied Science, Materials Science and Information Technologies in Industry* (2014), 1692–695.

[5] GroupLens. [n. d.]. GroupLens Research lab. http://grouplens.org/

[6] Mihwa Han. 2017. Bias in Movie Ratings. https://mihwa-han.github.io/project2.html

[7] Nicolas Hug. [n. d.]. Surprise. https://surprise.readthedocs.io/en/stable/index.html

[8] Ke Ji and Hong Shen. 2015. Addressing Cold-start: Scalable Recommendation with Tags and Keywords. 83 (2015), 42–50. https://doi.org/10.1016/j.knosys.2015.03.008.

[9] Rajaraman A. Leskovec, J. and J. Ullman. 2016. *Mining of massive datasets* (2nd ed.). Delhi: Cambridge University Press.

[10] GroupLens Research. [n. d.]. MovieLens 20M Dataset. https://grouplens.org/datasets/movielens/