University of New South Wales

COMP9417 - Report

Recommender system using collaborative filtering

Xuanxin Fang	z5142897
Tian Luan	z5041134

August, 2019

Contents

1 Introduction

Recommender systems are widely considered useful to predict user preferences and give recommendations. They can be utilized in many fields, especially for commercial purposes. In the past, researchers have developed many algorithms and models for it.

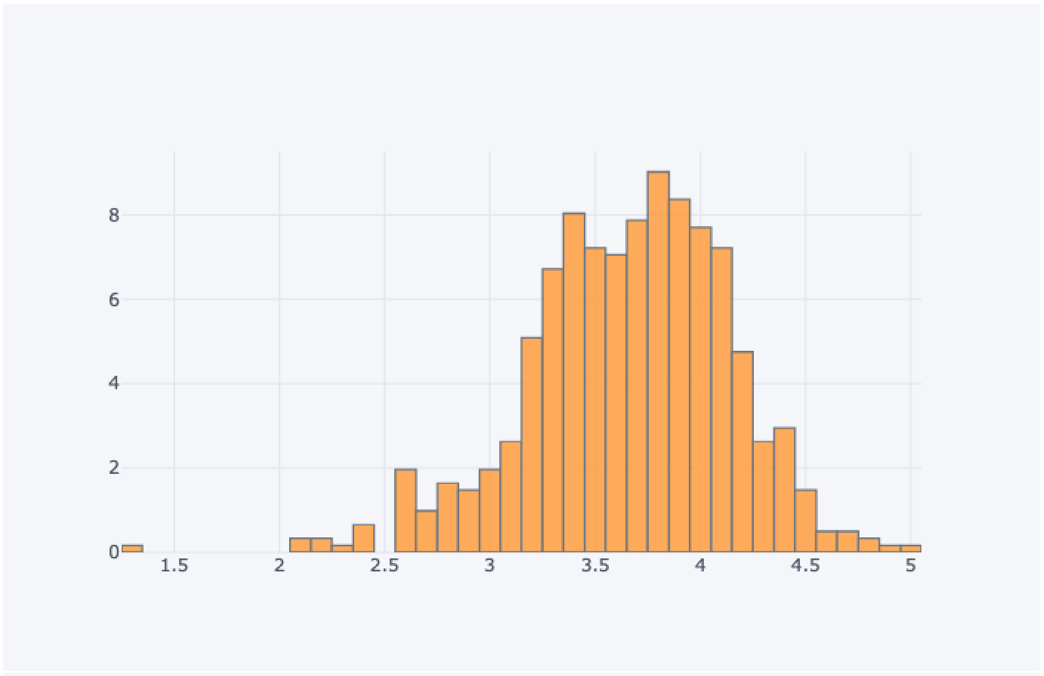
In this report, we built a recommender system for movies. We used movie rating data, applied various algorithms and evaluate the accuracy.

2 Data

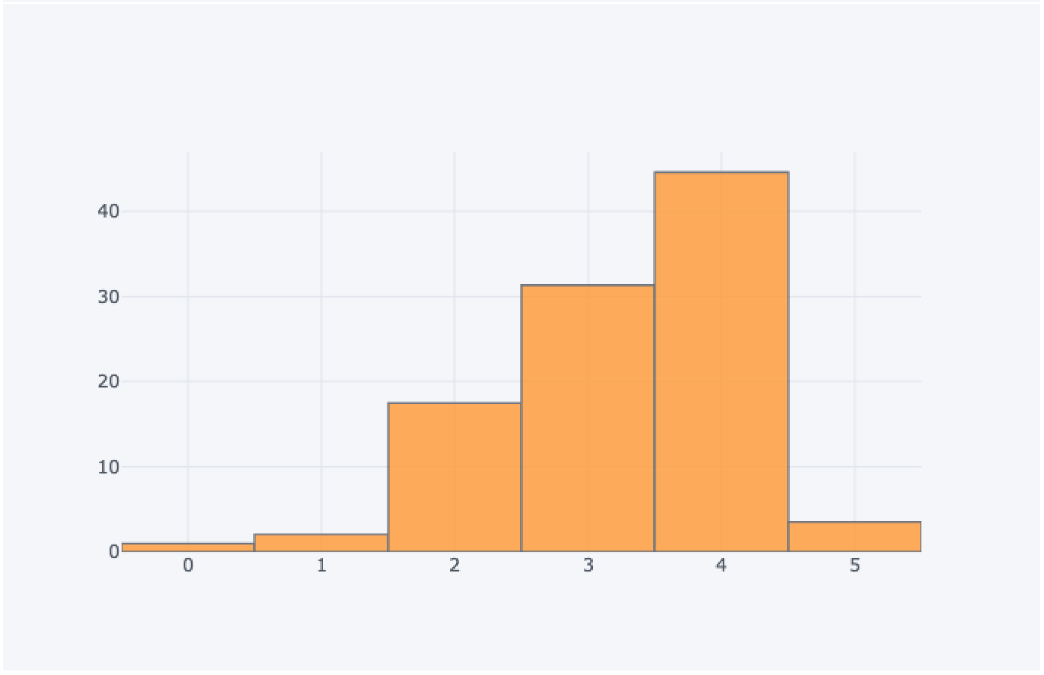
The raw data was from a collection *GroupLens data set 100k*. It contains 100836 ratings and 3683 tag applications across 9742 movies. We used *Matplotlib* and *seaborn* to visualize data distribution.

The first figure shows average rating for users. The second figure show average rating for movies.

We found the distributions of both average rating per user and movie are close to Gaussion distribution.



8
6
4
2
0
1.5
2
2.5
3
3.5
4
4.5
5



40
30
20
10
0
0
1
2
3
4
5

A rating matrix $rating\_M$ was transformed from the raw data. The rows were *userId*s, columns were *movieIds*. No rating was replaced by 0. Example is shown below.

$$rating\_M = \left[\begin{array}{llllllll} & & movieId & & & & \\ & & \mathbf{0} & \mathbf{1} & \mathbf{2} & ... & \mathbf{9723} \\ userId & \mathbf{0} & 4.0 & 3.0 & 3.0 & ... & 5.0 \\ & \mathbf{1} & 2.0 & 0 & 0 & ... & 1.0 \\ & \mathbf{2} & 0 & 3.0 & 4.0 & ... & 0 \\ & ... & & & & & \\ & \mathbf{609} & 0 & 2.0 & 0 & ... & 3.0 \end{array}\right] \quad (1)$$

The matrix has size 610×9724.

The sparsity was 1.70% which means 1.70% user-movie rating has a value.

3 Algorithms

Collaborative filtering is a common technique used to generate recommendations to users. It can be based on explicit or implicit user ratings.[3] Explicit rating is the scores that users give to products. For example, 1 - 5 score as movie rating. Implicit rating could be implied by user behaviours. For example, clicks or views. In this project, explicit rating was used as the movie rating was given.

Collaborative filtering has three types: Memory-based, Model-based, and Hybrid.[1] In this report, the first one and second one were used and discussed.

3.1 Memory Based Collaborative Filtering

Both user-based and item-based approaches are memory based.

3.1.1 User Based

Let $sim(u, u')$ be the similarity between target user u and another user u'. Then, in user-based collaborative filtering, [5]

$$sim(u,u') = rating_M.dot(rating_M.T) + \epsilon$$

Then, we normalize it

$$norms = sqrt(diagnol(sim(u,u')))$$

$$sim(u,u') = sim(u,u')/norms/norm.T$$

Let r_{ui} be the rating that user u gave to movie i. We assume r_{ui} is related to other users' rating to the given movie $r_{u'i}$. The more the target user is similar to the second user, the more the second user's rating is important. So we summed up all other users' ratings and gave them weights based on their similarity to target user. So the predicted rating from user u for movie i is [5] :

$$\hat{r_{ui}} = \frac{\sum_{u'} sim(u,u') r_{u'}}{\sum_{u'} |sim(u,u')|}$$

After the prediction, we evaluated the prediction against actual values. *Mean Square Error* was used to measure difference between the predicted rating $\hat{r}$ and the actual rating r. [4]

$$MSE = \frac{1}{N}\sum_{i}^{N}(r_i - \hat{r}_i)^2$$

The error rate was 3.15.

3.1.2 Item Based

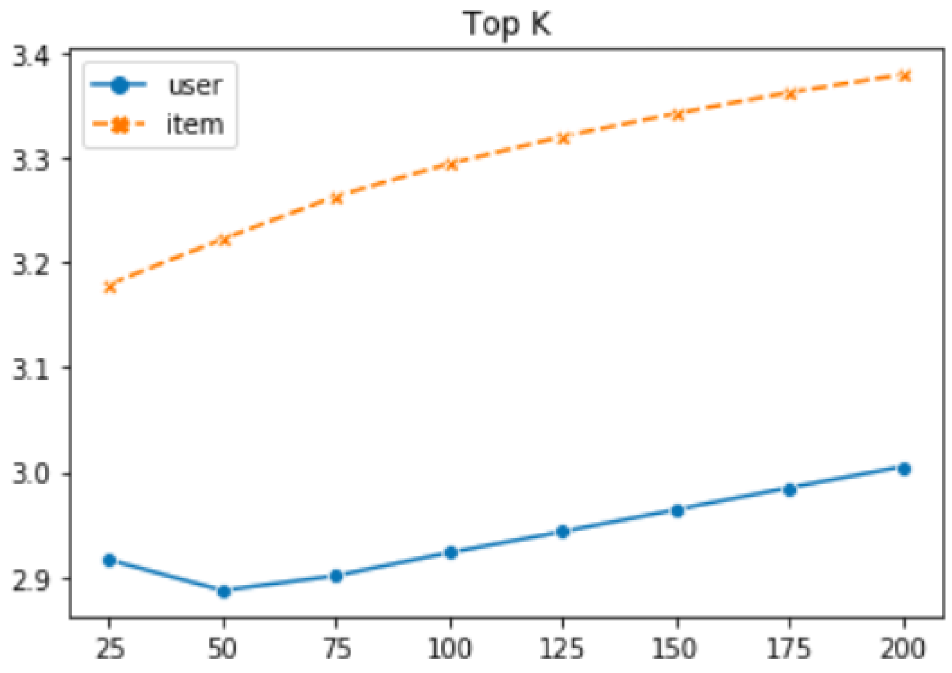
The idea for item-based is very similar to user-based. Other movie ratings from the same user could imply the rating for the target movie. The similarity matrix is between target movie and other movies rated by the same user. So the predicted rating for movie i from user u is:

$$\hat{r_{iu}} = \frac{\sum_{i'} sim(i,i') r_{i'}}{\sum_{i'} |sim(i,i')|}$$

Again, *Mean Square Error* was used to evaluate the prediction. The error rate was 3.69. It is slightly greater than user-based.

3.1.3 Top K

To optimize the result, parameter k was added. Only k most related users or items were selected to predict the rating.



As shown in the figure above, top K method did decrease the error rate. In our opinion, the reason was that top K method filtered out noises that were not very related to the target user or item.

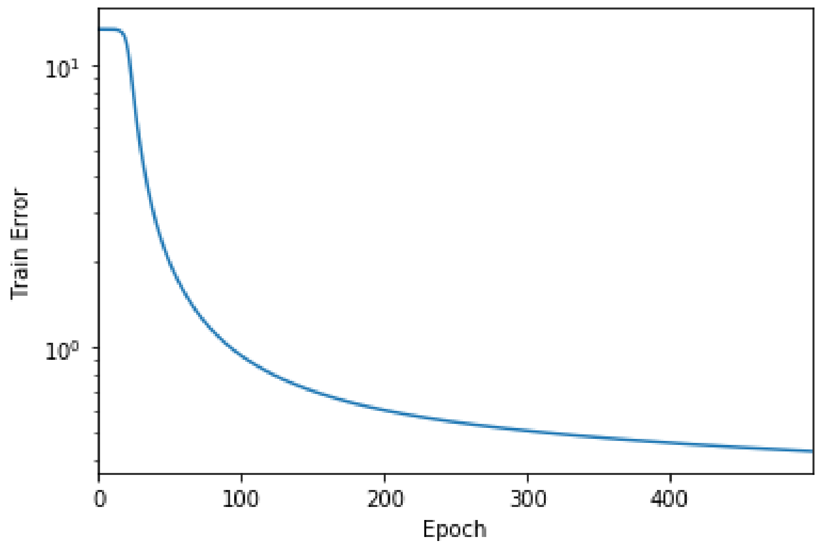
Given limited time, we set $k = 40$ as some articles suggested (e.g [5]). However, further experiments on the value of k should be conducted to find the optimal value. As k varies, the error rates will be different.

3.2 Model Based Collaborative Filtering

3.2.1 Matrix Factorization

One popular model-based approach is called Matrix Factorization.[6] The main idea[6] of this approach is to find representations of both users and items in a lower dimensional latent space. For example, we could encode each movie in vector form, which could imply meanings like movie length, genres and etc. More specifically, a user can be represented as $U_i \in R^{1 \times latentfactor}$, which means the ith user has a latent vector U_i. For each item, we have $V_j \in R^{latentfactor \times 1}$. Hence the rating of movie j given by user i can be calculated as the inner product $R_{ij} = U_i V_j$. In this case, for a given matrix $R^{users \times items}$, we need to find latent matrix U and V.

There are many ways to achieve that. In this report, we implemented matrix factorization through neural network framework. Parameters were initialized randomly. Estimated ratings $\hat{Rij} = U_i V_j$ were calculated. Mean square error (formula described in the former seciton) was used as the loss function. Stochastic gradient descent was the optimiser. The aim is to minimize loss on training dataset.

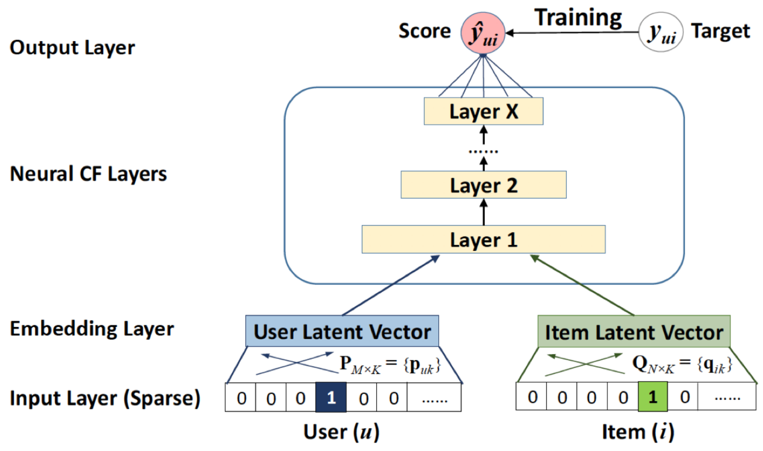


After 500 epochs, loss converges. On testing dataset, the RMSE was: 1.115.

3.2.2 Multi-layer Perceptron

Matrix fatorization performs better than simple memory based models. However, it is deemed as a linear model of latent factors. We could introduce some non-linear methods such as perceptron. Hence, we implemented a multi-layer perceptron to check if non-linear could do better than MF model.

Neural network framework [2] is shown below.



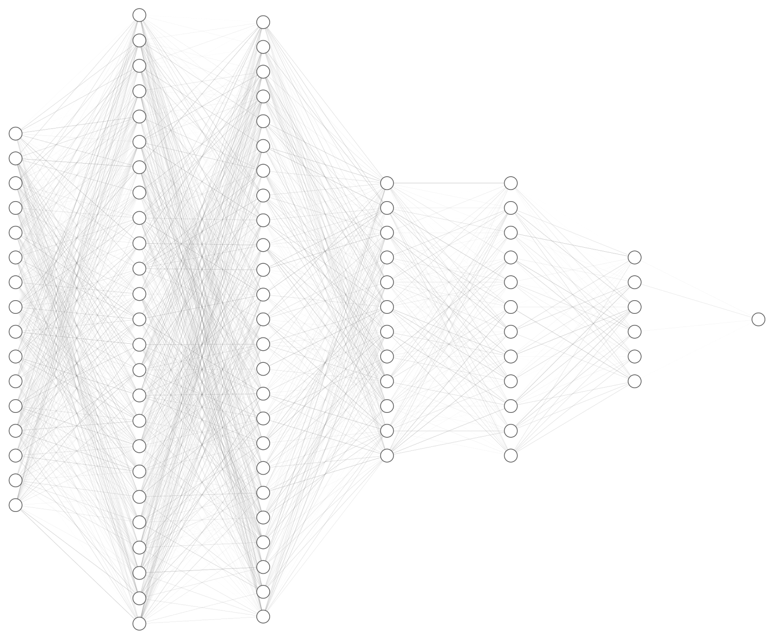
The architecture can be divided into three parts[2].

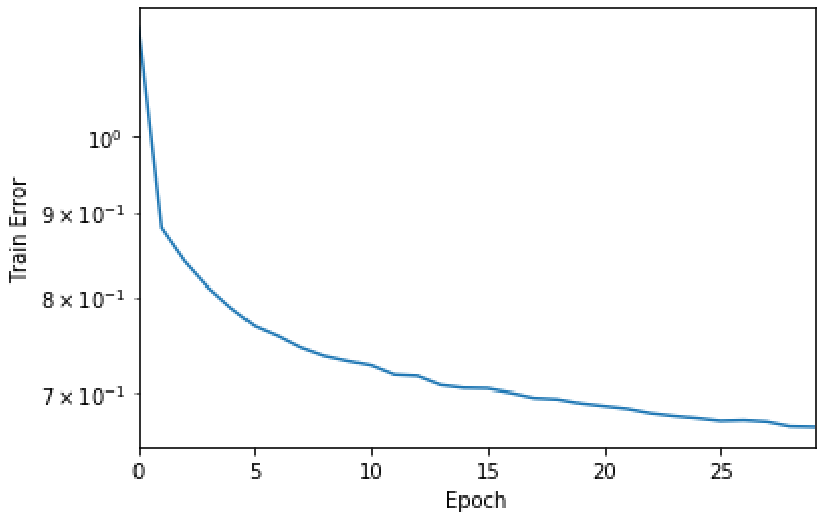
1. Input and embedding. For both users and items, encoding separately to a [input$size$, $latent$factor] matrix. Then concatenate to form an input matrix

2. Hidden layers. There are many architectures of hidden network. We implemented a basic multi-layer perceptron.

3. Output layer. Prediction should be a non-negative number, so that we use ReLU as activation function. During training process, we chose Adam optimizer to minimize the mean squared error between predicted ratings and target ratings.

The brief architecture of hidden layers was shown below, two hidden layers, each followed by a batch normalization layer. On this bases, to avoid over-fitting, we added a dropout layer after each hidden layer.

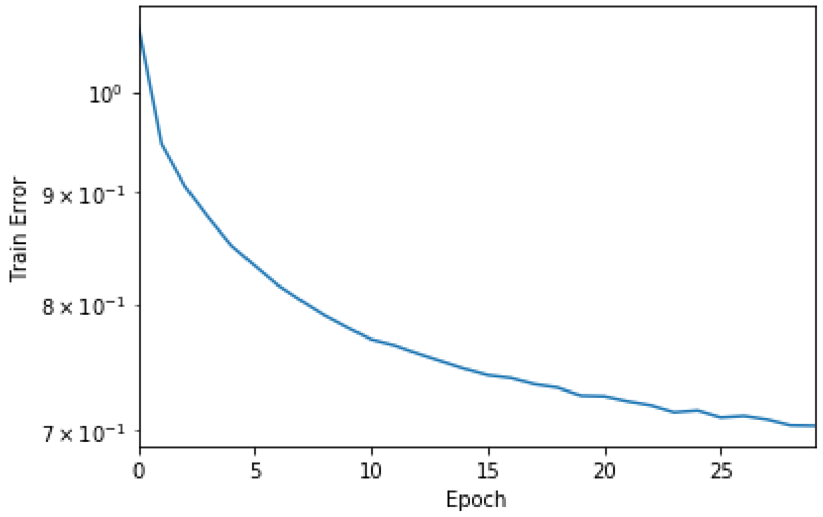




After 25 epochs, loss converged and RMSE was 0.882

3.2.3 Hybrid: MF-MLP

Since we have a linear model(MF) and a non-linear model(MLP), it is interesting to test if the fusion of MF and MLP could performs better[2]. We took the last layer of each model and concatenate them to form a MF-MLP layer. Then they are mapped to an extra hidden space before final prediction.



Loss function converged after 30 epochs and the RMSE was 0.866.

4 Summary

For user-base and item-based collaborative filtering algorithms, we think user-based is slightly better as the error rate was smaller. Top k selection can decrease the error rate. For user-based, $k = 50$ gave the lowest error rate, 2.887, while for item-based, $k = 25$ gave the lowest error rate, 3.179, Further tuning on parameter k could be done in the future.

For three model-based models, we found the Multi-layer Perceptron and MF-MLP were slightly more accurate and converged faster than Matrix Factorization. Also, the hyper-parameters for those two had bigger impacts on the results. MF-MLP was not more accurate than Multi-layer Perceptron.

In our experiments, the dropout layers could decrease over-fitting for Multi-layer Perceptron and MF-MLP. We found 0.3 was the best setting as dropout rate.

Overall, we compared the most simple neural networks with memory-based model. We found neural networks had a significant advantage. We thought if we used more features from the data and applied more complex neural networks, the error rate would be even lower.

References

[1] *Collaborative filtering.* URL: https://en.wikipedia.org/wiki/Collaborative_filtering#Memory-based.

[2] Xiangnan He et al. "Neural Collaborative Filtering". In: *Proceedings of the 26th International Conference on World Wide Web - WWW fffdfffdfffd17* (2017). DOI: 10.1145/3038912.3052569. URL: http://dx.doi.org/10.1145/3038912.3052569.

[3] Shuyu Luo. *Introduction to Recommender System.* 2018. URL: https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26.

[4] *Mean Square Error.* URL: https://en.wikipedia.org/wiki/Mean_squared_error.

[5] Ethan Rosenthal. *Intro to Collaborative Filtering.* URL: https://www.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/.

[6] Wikipedia contributors. *Matrix factorization (recommender systems) — Wikipedia, The Free Encyclopedia.* [Online; accessed 11-August-2019]. 2019. URL: https://en.wikipedia.org/w/index.php?title=Matrix_factorization_(recommender_systems)&oldid=901580528.