Performance Analysis of Collaborative Filtering Techniques

TZE-QIAN ENG[1], MING-HUANG LIN[2], JIE-YAO TANG[3]

Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

[1]m11015802@ntust.edu.tw

[2]m11015021@ntust.edu.tw

[3]m11015080@ntust.edu.tw

Abstract

In recent years, recommendation system has been used widely by most companies to predict customers' preferences based on their past behavior and also user profiles. The recommendation system benefits both the companies and customers by helping users to discover new and relevant products as well as increasing the revenue of the companies. There are different recommendation techniques, such as content-based filtering, collaborative filtering, and hybrid filtering. This paper focuses on the evaluation of different methods used in collaborative filtering, by comparing their performance on a dataset called MovieLens 100K dataset. The methods that were used in the experiment are namely matrix factorization based and neural-based models.

Keywords

Recommendation System, Collaborative Filtering. Matrix Factorization

I. INTRODUCTION

Recommendation system has been widely adopted by most companies in their products, ranging from online services such as e-commerce and social media sites to hand-held devices including tablets, smartphones, etc. A recommendation system that largely depends on users' behavior and past interactions to predict the user preferences on a particular item(s), is known as collaborative filtering.

Collaborative filtering techniques can be categorized into 2 categories, namely the memory-based approach and the model-based approach (our focus in this paper). In a memory-based approach, the main idea is to recommend items based on similarity, there is no learning of parameter needed. Usually, the similarity is calculated using Cosine similarity or Pearson correlation coefficients.

Whereas among various techniques in model-based approach, matrix factorization (MF) is the most popular and commonly used, which projects the representation of a user and an item, called *latent factor* into one single latent space, and perform dot product of the two vectors. The latent factors are the parameters learned in collaborative filtering using gradient descent or any other optimization algorithms.

$$cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2}\sqrt{\sum_i B_i^2}}$$

Figure 1: Cosine similarity formula

Besides matrix factorization, the recent state-of-the-art technique in collaborative filtering is neural collaborative filtering (NCF) [1], which we will discuss further in the next section.

This paper will study the performance between MF and NCF. Section 2 will discuss the techniques that were used in the experiment in detail, and section 3 will describe the details of the experiment. In section 4 the results of the experiment will be discussed. Section 5 will discuss future work as well as the conclusion.

II. PRELIMINARIES

In this section, we review the existing solution, which is the techniques mentioned in section 1. Since our work is not to propose a new solution, we do not formulate a problem here.

A. Matrix Factorization

In matrix factorization, the main concept is to decompose or break down a sparse user-item rating matrix into two lower dimensionality matrices, representing the user and item called latent factors. Then, the dot product is applied to these two latent factors to reconstruct a new rating matrix, so that the original sparse user-item matrix is now turned into a dense matrix. Then, a loss function is used to calculate the loss between the actual rating and the predicted ratings, which is the new rating matrix. Finally, an optimizer is used to calculate and step down or up the latent factors to get a lower loss, so that the predictions will get better and better.

There are several methods of matrix factorization and for our experiment, we chose the

popular Funk SVD, which won the Netflix prize competition. The Funk SVD method uses the Sum of Squared Error as the loss function and Stochastic Gradient Descent (SGD) as the optimizer. In addition, to make the model generalized well and avoid overfitting, a penalty term or regularization is implemented to the loss function.

In our experiment, we used another matrix factorization method called the *EmbeddingDotBias* model, which is implemented by the fastai library. In this method, we can define our number of latent factors, then 2 embeddings with length same as the number of latent factors are created representing the user and item. The embeddings parameter are randomly initiated and later learned by the model. Similar to the Funk SVD, L2 regularization and SGD optimization are implemented as well.

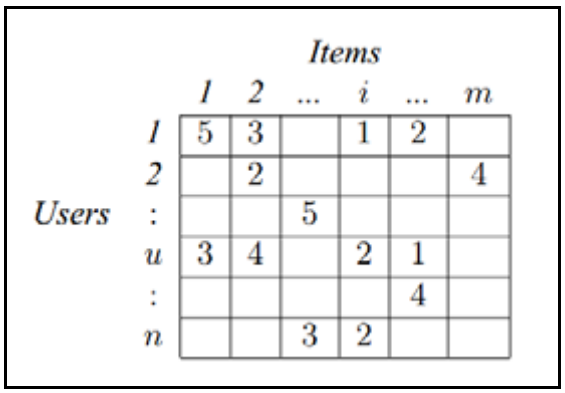

Figure 3: Sparse user-item matrix

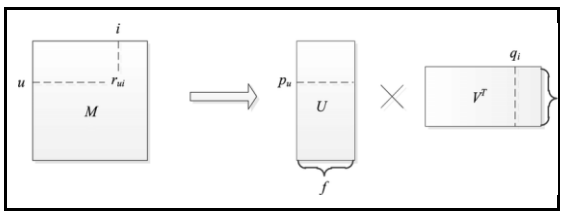

Figure 4: Funk-SVD factorization model

B. Neural Collaborative Filtering

The model proposed in [1] which will be implemented in this experiment is actually based on two different networks that are combined into one. The two networks are the MF and multilayer perceptron network (MLP). The idea is to make use of the advantages of both the linearity from MF and non-linearity from MLP.

MLP is the simplest kind of deep neural network (DNN), with at least one hidden layer of neurons and each of them is later on connected to a non-linear activation function. Each layer in MLP is densely connected, which means that the neuron in the previous layer is connected to every neuron in the next layer.

Finally, the MF and MLP networks are combined together into an NCF model called NeuMF. Basically, this model is the result of the ensemble from both MF and MLP. Because of this change, it improves the performance of linearity of MF and non-linearity of MLP for the model of the user-item latent structure. The original loss function and optimization used in NCF are the binary cross-entropy and adam optimizer.

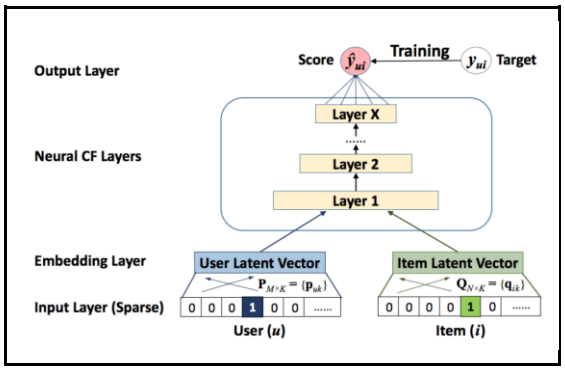

Figure 5: NCF architecture

The NCF architecture is divided into four parts.
1. Input Layer binarise a sparse vector for a user and item identification
2. Embedding layer is a fully connected layer that projects the sparse representation to a dense vector.
3. Neural CF layers use MLP combined with MF to map the latent vectors to prediction scores.
4. The final output layer returns the predicted score by minimizing the pointwise loss/pairwise loss.

III. EXPERIMENT SETTINGS

In this section, we conduct the experiments on the three collaborative filtering models mentioned in section 2, with the aim to compare the performance of the models.

A. Dataset

We carried out the experiments with the public accessible dataset: MovieLens. The MovieLens dataset is a movie rating dataset that has been used widely to evaluate the performance of collaborative filtering algorithms. In particular, we used the MovieLens-100K dataset, which consists of 100K ratings from 943 users and 1682 movies. The reason we choose this dataset is there exist other datasets such as 1M, 10M, 20M, etc, which is too large and time-consuming for training the model.

B. Training and Evaluation

In our experiments, we fix the experiment settings, such as the loss and optimizer, to ensure that this is a meaningful comparison, although it is

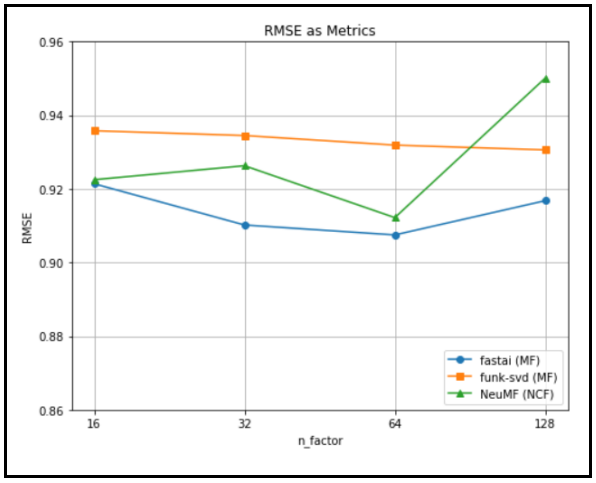

Figure 6: Number of factors vs RMSE

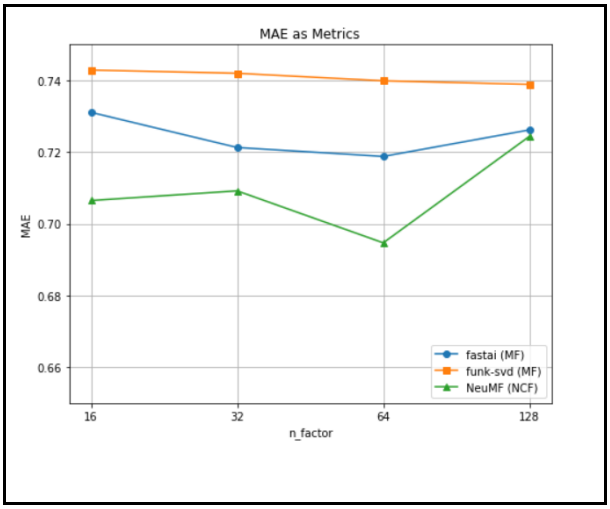

Figure 7: Number of factors vs MAE

Factors	FastAI (MF)		Funk-SVD (MF)		NeuMF (CNF)	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
16	**0.921**	0.7311	0.936	0.7429	0.923	**0.7065**
32	**0.910**	0.7213	0.935	0.7420	0.926	**0.7092**
64	**0.908**	0.7188	0.932	0.7399	0.912	**0.6947**
128	**0.917**	0.7262	0.931	0.7389	0.950	**0.7244**

Table 1:

possible that different settings might lead to even better results. We select mean squared error (MSE) with L2 regularization and a simple SGD as an optimizer for all of the methods. The evaluation metrics used are the root mean squared error (RMSE) and mean absolute error (MAE), these two metrics are chosen as the models are predicting the user ratings which range from 0 to 5.

For the training process, we set 50 epochs with early stopping, a set of latent factors number, d = {16,32,64,128}, a learning rate of 0.01, and a weight decay rate of 0.2.

IV. RESULTS

The results are reported in Figures 6 and 7. Based on the findings, we can see that one of the matrix factorization methods, Funk-SVD performs the worst across all evaluation metrics and all dimensions of latent factors except one. Whereas for NCF and fastai, these two models outperform each other in different evaluation metrics.

We can see that when the number of latent factor increase, the evaluation metrics decrease until one point where the number of latent factors is the highest. We assume a higher number of latent factors may lead to the overfitting of the model since our dataset is not large enough.

Although it seems that the result obtained is less significant and has a small difference, in collaborative filtering competitions such as the Netflix Prize, a difference in RMSE of 0.01 is considered very large [2]. One example would be in the Movielens 10M dataset, it took the community about 4 years to lower the RMSE from 0.7815 to 0.7634.

Based on the findings in [1], the performance of NCF is better than MF across all evaluation metrics. One possible reason we got different results would be caused by the dataset and experiment settings. Since the MovieLens 100K dataset is much smaller than the others, the results would be significant if a larger dataset is used in the experiments. As mentioned before, since we want to produce a meaningful comparison, we standardized the experiment settings.

V. CONCLUSION

In our works, we studied the performance of different collaborative filtering methods, mainly between Matrix Factorization and Neural Collaborative Filtering. From our results, MF and NCF would outperform in some cases, although the settings do not show any evidence that MF is superior. As far as future work is concerned, the next

step would be increasing the quality of experiments by evaluating using different metrics, like in [1], and also using larger datasets for training.

References

[1] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173-182).

[2] Rendle, S., Krichene, W., Zhang, L., & Anderson, J. (2020, September). Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM Conference on Recommender Systems* (pp. 240-248).