Top-N Recommendation Model Based on SDAE

Rui Bao*, and Yipin Sun

1National University of Defense Technology, Changsha, Hunan

* E-mail: mr_rayleigh@163.com

Abstract. Collaborative filtering is the mainstream approach to personalized recommendation. For the cold start problem faced in collaborative filtering, it is a hot research topic to introduce the user's side information into the recommendation model. Different from the matrix decomposition idea adopted in the existing methods, we propose a Top-N recommendation model using the side information of the user based on the reconstruction function of the stacked denoising auto-encoder. Experimental results show that the model outperforms the existing method in Recall. In addition, we explore the influence of missing ratings and user side information vector into the loss computation. The experimental results show that ignoring the missing ratings in the loss function is beneficial to improve the performance of the model.

1. Introduction

Collaborative filtering is the most popular recommendation algorithm for personalized recommendation. It mainly performs recommendation tasks based on the user-item rating matrix, such as the item-based collaborative filtering recommendation algorithm on Amazon [1], and the matrix factorization algorithm that won the Netflix Prize [2, 3]. The core data of collaborative filtering is the user-item rating matrix. The cold start problem is a great challenge that collaborative filtering faces. For new users, the model cannot make accurate recommendations for them. For new items, the model cannot recommend them [4]. In the real world, we may get some side information about the user, such as the demographic information, which is an important way to alleviate the user's cold start problem [15].

At present, the application of side information in collaborative filtering generally adopts the idea of matrix factorization [5, 6, 16]. Collective Matrix Factorization (CMF) [5] applies these side information to traditional matrix factorization, while aSDAE (additional Stacked Denoising Auto-Encoder) [6] utilizes the compression function of stacked denoising auto-encoder to obtain latent factor model for matrix factorization, which helps to improve the robustness and performance of the model [7, 8].

In this paper, we propose a recommendation model based on stacked denoising auto-encoder from the perspective of data reconstruction. The model reconstructs the input rating vector and the user's side information vector with a stacked denoising auto-encoder, and selects N items for recommendation from the output. Experimental results show that the model proposed in this paper is superior to CMF, the model based on aSDAE [6] in Recall. In addition, we analyze and evaluate the impact of some important factors in our model:

- In the model input, we explore two import methods of dealing with the missing rating values, treating the missing value as 0 or ignoring the missing value directly in the objective function.
- In the calculation of the model loss function, whether the user side information vector...
participates in the operation has an effect on model performance.

- Whether the dimensional size of the compressed vector has an effect on model performance.

We organize the rest of the paper as follows. In Section 2, we introduce the definition of the problem, related notations and some basic concepts. Section 3 describes our proposed model and some details. In Section 4, we show the experimental settings and results, compare the performance resulted by different factors. Then compare with other models. The conclusion is in Section 5, in which we also discuss some future work.

2. RELATED WORK

2.1 Problem Definition

\( U \) represents the set of all users \( U = \{u | u = 1, 2, 3, \ldots, m \} \). \( I \) represents the set of all products \( I = \{i | i = 1, 2, 3, \ldots, n \} \). \( R \) represents a user-item rating matrix, where each row represents a user's rating vector, each column represents the rated vector of an item, and each entry \( r_{ui} \) in \( R \) corresponds to user \( u \)'s rating on item \( i \). Here we use explicit rating. That means that the user's rating directly represents the user's preference on item. For example, he rating scale in MovieLens is 1-5 [16]. The higher the score indicates the higher the user's preference for the item. Implicit ratings have no obvious user preference, such as purchase records, and the corresponding item ratings are unified to 1 [18]. \( \hat{R} \) and \( \hat{r}_{ui} \) mean the predicted rating matrix and rating value respectively. Most collaborative filtering methods focus on obtaining accurate predicted rating matrix.

Usually, we can get some basic user information, such as age, etc. The users’ side information can be represented by matrix \( D \), \( d_u = (d_1, d_2, \ldots, d_i) \) represents the side information vector of user \( u \).

2.2 Denoising Auto-Encoder

Auto-encoder (AE) [9] is a neural network and an important way of unsupervised learning. Basic auto-encoder [21] consists of 3 layers, the input layer, one hidden layer and the output layer. In the actual situation, the input data is often accompanied by noise. Therefore, to improve the robustness of the auto-encoder, we add noise to the input data constructing a corrupted version of input vector, which results in Denoising Auto-Encoder (DAE) as shown in Figure 1.

![Figure 1: The schematic diagram of DAE.](image)

\( x \) represents the original input, \( z \) is the hidden layer output, and \( y \) is the final output. There are usually two ways to add noise [10]. One is additive Gaussian noise, adding Gaussian noise to the original input, which is composed of Gaussian noise added to the entire input vector. This is defined
as:

\[ \tilde{x} = x + \alpha \cdot G, G \sim N(\mu, \sigma^2) , \]

where, \( \alpha \) is the noise factor. The other is mask-out/drop-out noise \[10\], that is, each item of the vector is set to 0 with a certain probability of q, as shown below

\[ P(\tilde{x}_i = 0) = q, \]
\[ P(\tilde{x}_i = \delta x_i) = 1 - q, \]

where, \( \delta = \frac{1}{1 - q} \).

The auto-encoder has two important functions. The encoding function, from the input layer to the hidden layer, can be used to compress vector. The decoding function, from the hidden layer to the output layer, is used to reconstruct the input vector from the compressed vector. The encoding part is defined as :

\[ z = f(W_h \tilde{x} + b_h), \]

where, \( W_h \) indicates weight matrix, \( b_h \) represents bias part. The decoding part is defined as:

\[ y = g(W_o z + b_o). \]

The loss function of DAE is commonly the mean squared error between output \( y \) and original input \( x \).

2.3 Related Models

Collective Matrix Factorization (CMF) is a collaborative filtering recommendation model with side information applied \[5\]. It decomposes the user-item rating matrix, the user side information matrix and the item side information matrix separately to obtain the latent factor models of users and items. The core idea is still to decompose the user-item scoring matrix \( R \) into matrices \( P \) and \( Q \), except that \( P \) and \( Q \) also participate in the decomposition of the user side information matrix and the item side information matrix, respectively.

The hybrid model based on aSDAE (additional Stacked Denoising Auto-Encoder) \[6\] is somewhat similar to the CMF model. The difference is that the former uses the encoding function of the stacked auto-encoder to obtain the latent factor model instead of matrix factorization. After get the latent factor vector \( p_i \) of user \( i \) and \( q_j \) of item \( j \), the estimated rating \( \hat{r}_{ij} \) is the inner product of \( p_i \) and \( q_j \).

3. PROPOSED METHODOLOGY

3.1 Model Proposed

A stacked auto-encoder can be seen as multiple auto-encoders stacked together, containing multiple hidden layers. Research has shown that multiple hidden layers are beneficial to feature extraction and ultimately improve the performance of neural networks \[8\]. We also know that the denoising auto-encoder has stronger robustness than the basic auto-encoder. Inspired by AutoRec \[11\] and CDAE \[10\], we build a recommendation model based on the reconstruction ability of the stacked denoising auto-encoder reconstruction. At the same time, in order to alleviate the user's cold start problem, the user's side information vector is used as an extended input and participates in the calculation of the model, as shown in Figure 2.
The input layer is composed of two vectors, $\tilde{r}_u$ represents the corrupted rating vector of user $u$, and $\tilde{d}_u$ represents the corrupted version of the user side information vector. Suppose there are altogether $L$ hidden layers. The hidden layer $l$ is calculated as

$$h_l = f(W_l h_{l-1} + b_l),$$

where, $h_0 = [\tilde{r}_u, \tilde{d}_u]$, it means concatenate $\tilde{r}_u$ and $\tilde{d}_u$. The output layer is calculated as

$$[\hat{r}_u, \hat{d}_u] = g(W_{L+1} h_L + b_{L+1})$$

### 3.2 Objective Function

In recommendation algorithms, the objective functions can be divided into two classes, point-wise and pair-wise [10, 12, 17]. Point-wise is designed to measure the difference between the actual rating and the predicted rating. Pair-wise aims to measure the difference between the difference of a pair of actual ratings and the corresponding estimated rating difference.

In our model, since the user’s side information is used, it is inappropriate to use the pair-wise objective function and so we use the point-wise objective function.

$$\min \frac{1}{m} \sum_{i=1}^{m} \| [r_u, d_u] - [\hat{r}_u, \hat{d}_u] \|^2$$

In collaborative filtering, how to deal with missing ratings is an important issue. In matrix factorization, the objective function directly ignores the missing scoring items, as shown in the following equation.

$$\sum I_{ij} (r_{ij} - \hat{r}_{ij})$$

$I_{ij}$ is the indication function, equals to 1 when $r_{ij}$ is not missing, otherwise 0. You can use regularization in the formula according to the situation. For some models, you can also directly treat the missing rating as 0 when calculation. In our experiment, we try both methods and see the influence on performance.

In addition, since the real target of the model is the predicted rating vector $\hat{r}_u$, we don’t care about $\hat{d}_u$. We try to modify the loss function slightly, considering the difference between all input and output, or just the difference between actual rating vector and the predicted one. When only considering the difference between the score vector and the reconstructed score vector, the formula is as follow:

$$L([r_u, d_u], [\hat{r}_u, \hat{d}_u]) = \| r_u - \hat{r}_u \|^2$$

### 4. EXPERIMENTS

#### 4.1 Datasets
We use two datasets from different real-world domains, MovieLens-100k and MovieLens-1m, for our experiments. MovieLens [13] datasets are commonly used for recommender systems [6, 10, 18]. These datasets record users’ preference for movies, where users rate movies with a int score between 1 and 5, collected by GroupLens Research. The MovieLens datasets also provide users’ demographic information and movies’ tag information. The MovieLens-100k dataset contains 100K ratings from 943 users on 1682 movies, and the MovieLens-1m dataset contains about 1 million ratings from 6040 users on 3952 movies [10]. Moreover, each user rates more than 20 movies.

| Table 1: Statistics of Datasets. |
|---------------------------------|
| Datasets | #users | #items | density(%) |
|---------|--------|--------|------------|
| MovieLens-100k | 943    | 1682   | 6.30       |
| MovieLens-1m  | 6040   | 3952   | 4.19       |

4.2 Some Details

The optimizer we use in our model is Adam [14], which is widely used for neural networks. We set the number of hidden layers to 5. And the learning rate $\eta$ is 0.001.

We choose 20% ratings randomly from the datasets as testing data, the rest 80% as training data, then perform 5-fold cross validation and get the average result. We finally compare with existing model PMF, CMF, and hybrid model based on aSDAE on Recall. The data of these models come from the work of Xin Dong et al [6].

4.3 Evaluation Metrics

In our model, we select the $N$ items that have the highest predicted score but the user has not previously rated as a recommendation list. We use Recall and Mean Average Precision (MAP) [19, 20] to evaluate the recommendation.

Recall and Precision are defined as [10]:

\[
\text{Recall}_@N = \frac{|S_{N,\text{rec}} \cap S_{\text{adopted}}|}{|S_{N,\text{rec}}|},
\]

\[
\text{Precision}_@N = \frac{|S_{N,\text{rec}} \cap S_{\text{adopted}}|}{N},
\]

where, $S_{N,\text{rec}}$ indicates the recommendation list, $S_{\text{adopted}}$ indicates the list of item that the user actually rated in the testing data.

MAP comes from the information retrieval. It is the mean value of AP, which is used for evaluating a rank list. For Top-N recommendation, AP is defined as

\[
\text{AP}_@N = \frac{\sum_{k=1}^{N} \text{Precision}_@k \times \text{rel}(k)}{|S_{N,\text{rec}} \cap S_{\text{adopted}}|},
\]

where, rel($k$) is a indication function. When the item at rank $k$ in the recommendation list has been rated in the testing set, it is 1, otherwise 0. MAP here is calculated as the average of all users’ AP. Like Yao Wu’s work [10], we mainly show the performance Recall$_@N$, MAP$_@N$ with $N = \{5, 10, 50\}$.

4.4 Experimental Results

At first we explore the influence whether or not considering the missing ratings in computing loss function. Figure 3 shows the result of our model on MovieLens-100k dataset, Figure 4 on MovieLens-1m dataset.
Figure 3: Performance of whether or not to consider missing ratings in loss function on MovieLens-100k.

Figure 4: Performance of whether or not to consider missing ratings in loss function on MovieLens-1m.

From Figure 3 and Figure 4, we may know that not considering the missing values in loss function does influence the performance of Recall and MAP. It seems that not computing missing ratings in loss function could improve the recommendation performance. More accurately, the improvement on MAP is generally greater than on Recall.

Figure 5 and Figure 6 show the performance of whether the side information vector is used for computing loss function. We find that whether to use user side information in computing loss function has little influence on Recall and MAP. We wonder if it is because that in the datasets, user side information vector is too short compared to the rating vector.

Figure 5: Performance of whether to use side information vector when computing loss function on MovieLens-100k.
We then explore whether the encoding dimension has influence on model performance, the results are shown in Figure 7 and Figure 8. From the results we could make the conclusion that encoding dimension has little impact on model performance, we guess this is because we do not use the encoding vector directly.

Finally, we compare our model with other existing models on Recall. other models’ come from the work of Xin Dong [6]. The result is shown in Figure 9. From the figure we could say that our model outperforms other models which utilize side information on Recall.
5. CONCLUSION

In this paper, we present a model based on stacked denoising auto-encoder for Top-N recommendation using user side information. The use of side information helps to alleviate the user cold start problem in collaborative filtering. We conduct experiments on MovieLens datasets. We first explore some factors’ influence in our model, then compare our model with other models. The experimental results show that our model outperforms other model on Recall. In the future, we will try to investigate other model for better performance, especially deep learning models. Or we will try some new parts in our model, for example, replace the point-wise objective function with pair-wise objective function, and try to utilize item side information.

REFERENCE

[1] Linden, G., Smith, B., York, J. (2003) Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing, 7: 76-80.
[2] Koren, Y. (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Las Vegas. pp. 426-434.
[3] Koren, Y., Bell, R.M., Volinsky, C. (2009) Matrix Factorization Techniques for Recommender Systems. Computer, 42: 30-37.
[4] Adomavicius, G., Tuzhilin, A. (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge & Data Engineering, 17: 734-749.
[5] Singh, A. P., Kumar, G., Gupta, R. (2008) Relational learning via collective matrix factorization. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Las Vegas. pp. 650-658.
[6] Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., Zhang, F. (2017) A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems. In: AAAI. San Francisco. pp. 1309-1315.
[7] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P. (2008) Extracting and composing robust features with denoising autoencoders. In: ICML. Helsinki. pp. 1096–1103.
[8] Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y. (2011) Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In: ICML. Washington. pp. 833-840.
[9] Liou, C., Cheng, W., Liou, J., Liou, D. (2014) Autoencoder for words. Neurocomputing, 139: 84-96.
[10] Wu, Y., DuBois, C., Zheng, A.X., Ester, M. (2016) Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In: WSDM. San Francisco. pp. 153-162.
[11] Sedhain, S., Menon, A.K., Sanner, S., Xie, L. (2015) AutoRec: Autoencoders Meet Collaborative Filtering. In: WWW. Florence. pp. 111-112.
[12] Lee, J., Bengio, S., Kim, S., Lebanon, G., Singer, Y. (2014) Local collaborative ranking. In: WWW. Seoul. pp. 85-96.
[13] Harper, F.M., Konstan, J.A. (2015) The MovieLens Datasets: History and Context. TiSiS, 5: 19:1-19:19.
[14] Kingma, D.P., Ba, J. (2014) Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.
[15] Shi, Y., Larson, M., Hanjalic, A. (2014) Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. ACM Comput. Surv., 47: 3:1-3:45.
[16] Wu, C., Ahmed, A., Beutel, A., Smola, A.J., Jing, H. (2017) Recurrent Recommender Networks. In: WSDM. Cambridge. pp. 495-503.
[17] Zhuang, F., Luo, D., Yuan, N.J., Xie, X., He, Q. (2017) Representation Learning with Pair-wise Constraints for Collaborative Ranking. In: WSDM. Cambridge. pp. 567-575.
[18] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T. (2017) Neural Collaborative Filtering. In: WWW. Perth. pp. 173-182.
[19] Shani, G., Gunawardana, A. (2011) Evaluating Recommendation Systems. In: Shapira, B. (Eds) Recommender Systems Handbook. Springer US. pp. 257-297.
[20] Turpin, A., Scholer, F. (2006) User performance versus precision measures for simple search tasks. In: SIGIR. Washington. pp. 11-18.
[21] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H. (2006) Greedy Layer-Wise Training of Deep Networks. In: NIPS. Vancouver. pp. 153-160.