A Recommendation Approach Based on Latent Factors Prediction of Recurrent Neural Network

Ruihong Li[1,2](✉), Xingquan Zuo[1,2], Pan Wang[1,2], and Xinchao Zhao[3]

[1] School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
liruihongbob@qq.com, zuoxq@bupt.edu.cn

[2] Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China

[3] School of Science, Beijing University of Posts and Telecommunications, Beijing, China

Abstract. Recommender systems have received much attention due to their wide applications. Current recommender approaches typically recommend items to user based on the rating prediction. However, the predicted ratings cannot truly reflect users interests on items because the rating prediction is usually based on history data and does not consider the effect of time factor on uses interests (behaviors). In this paper, we propose a recommendation approach combining the matrix factorization and a recurrent neural network. In this approach, all the items rated by a user are considered as time series data. The matrix factorization is used to obtain latent vectors of those items. The recurrent neural network is taken as a time series prediction model and trained by the latent vectors of historical items, and then the trained model is used to predict the latent vector of the item to be recommended. Finally, a recommendation list is formed by mapping the latent vector into a set of items. Experimental results show that the proposed approach is able to produce an effective recommend list and outperforms those comparative approaches.

Keywords: Recommender systems · Matrix factorization · Recurrent neural network · Latent factor

1 Introduction

Recommender systems are received much attention in recent years [1]. They help users find information or goods that interest them and make enterprises get more profits by recommending suitable items to users. Recommender systems have been applied to a variety of areas, such as movies, music, books, news and services recommendation.

Numerous recommender approaches have been proposed over the years [2]. Among those approaches, content-based approaches and collaborative filtering (CF) approaches are two typical categories of recommend approaches.

CF approaches are probably the most successfully and widely used techniques in recommender systems and includes neighbor-based CF [1–4] and model-based CF.

As a representative of the model-based CF, the matrix factorization (MF) approach [5,6] represents the interaction between users and items with a rating matrix. It assumes that users and items are in the same latent space, and that each user or item can be represented by a latent vector in the latent space. It predicts unknown ratings in the matrix through a matrix factorization model, which decomposes the rating matrix into users and items latent vectors.

Many researchers proposed improved matrix factorization approaches to get better predicting results. Koren [7] proposed an approach combining the matrix factorization with a neighbor-based approach, and integrated implicit feedback in the modeling process. Salakhutdinov [8] proposed a probabilistic matrix factorization model which greatly reduce the overfitting of the traditional matrix factorization model. In [9], Salakhutdinov et al. further used the Markov Chain Monte Carlo model to optimize the parameters to reduce the overfitting and improve predicting accuracy of the probabilistic matrix factorization. In [10], the item factors between the matrix factorization and item embedding parts are shared, and the rating matrix and item co occurrence matrix are factorized at the same time. Some approaches are proposed to improve predicting results through adding supplementary information to MF [11–14]. For example, Gopalan et al. used Poisson factorization to model both user ratings and document content. Rather than modeling the two types of data as independent factorization problems, they connected the two latent factorizations using a correction term [11]. Mcauley et al. developed statistical models that combine latent dimensions in rating data with topics in review text, taking the corpus likelihood acts as a regularizer for the rating prediction model [12].

In practice, the predicted rating of a user on an item may not truly reflect the users interest on the item. For example, the rating of a user on an item is usually predicted based on all history data, and the predicted rating is not very high. That does not mean that the user does not have interest in the item all the time. The user may be interested in the item at some particular time because the interest of a user usually changes over time. Therefore, time is an import factor that has an effect on recommendation results.

In this paper, we propose a recommendation approach combining the matrix factorization approach with a recurrent neural network (RNN). Instead of predicting ratings of a user on items directly, our approach considers the items rated by a user as a time series data, and then use the RNN as a time series prediction model to predict a recommend list to the user. First, all items rated by a user are sorted according to the time when the user rates those items. Then, the matrix factorization approach is used to get latent factor vectors of those items. Subsequently, the RNN uses latent vectors of historical items to predict the latent vector of the item that may interest the user. Finally, a number of items with latent vectors closest to the predicted one forms the final recommendation list.

The main contributions of this paper include: (1) propose a recommendation approach combining the matrix factorization and the recurrent neural network; (2) latent factors of items are introduced into the prediction of a recommend list. To the best of our knowledge, our approach is the first work on applying latent factors to the time series prediction model for items recommendation.

This paper is organized as follows. Section 2 presents the proposed recommendation approach. Section 3 gives experimental results and the analysis. Section 4 concludes this paper.

2 Proposed Recommendation Approach

Given the time series of items rated by a user, our approach is to predict a set of items that the user will consume next.

Take the movie recommendation for example, suppose that there are n users $\{U_1, U_2, ..., U_n\}$ and m movies $\{I_1, I_2, ..., I_n\}$. All the movies watched by a user can be sorted by the time when the user watches movies, thereby getting a sequence of movies, denoted by S. Let $\{I_{t_k}, I_{t_{k+1}}, ...I_{t_{k+T}}\}$ be $(T+1)$ consecutive items in S, where t_k is the index of the kth item in S.

In our approach, items $\{I_{t_k}, I_{t_{k+1}}, ...I_{t_{k+T-1}}\}$ are used to predict item $I_{t_{k+T}}$. The key issue is how to express an item in the sequence S. A straight forward method is to adopt the one-hot encoder to represent an item [21]. However, the one-hot encoder would consume too many memory resources because there may be millions of items for some recommendation problems. In this paper, we use the matrix factorization approach to get a low-rank latent vector for an item. Each item is represented by a latent vector. The number of dimensions of the latent vector is much smaller than that of one-hot encoder of items.

Our approach is shown in Fig. 1. First, the matrix factorization approach is used to decompose the rating matrix into latent vectors of items, such that each item corresponds to a unique latent vector. Let κ be the number of dimensions of a latent vectors. In the example shown in Fig. 1, $\kappa = 5$. For the consecutive items $\{I_{t_k}, I_{t_{k+1}}, ...I_{t_{k+T}}\}$ in S, latent vectors of items $\{I_{t_k}, I_{t_{k+1}}, ...I_{t_{k+T-1}}\}$ are taken as the inputs of the RNN to predict the latent vector of item $I_{t_{k+T}}$. Finally, the recommend list is constructed by s items whose latent vectors are closest to the predicted latent vector. Suppose that n_j is the index of the jth item in the s sequence items. The recommendation list is expressed by $\{I_{n_1}, I_{n_2}, ..., I_{n_s}\}$. This approach integrates the time factor into the RNN model to emphasize the change of users interests and behaviors with time.

2.1 Generating Latent Vectors by Matrix Factorization

Matrix factorization is one of the most popular and useful CF models in recommender systems. Users interaction with items, especially explicit feedback, are typically represented by a rating matrix. Take a movie recommendation for example, a user rates a movie after watching it. The rating is from 1 to 5, representing the users evaluation on the movie. Figure 2 is a rating matrix with m

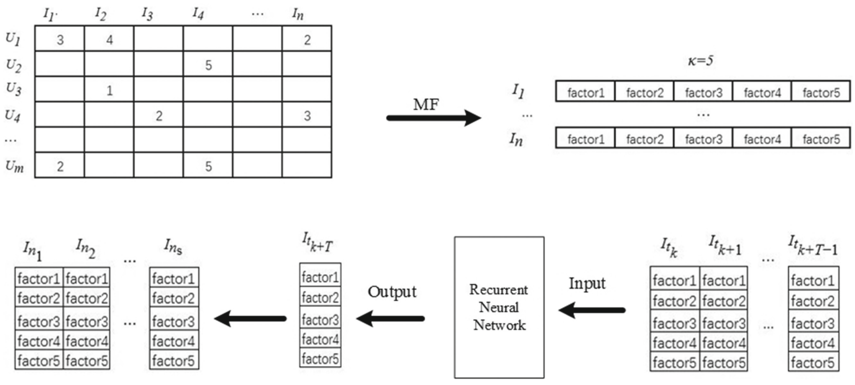
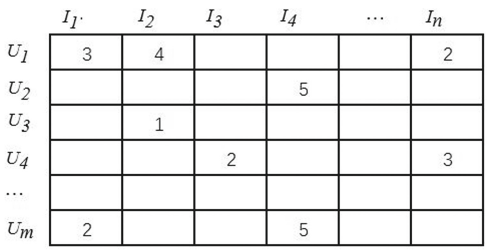


Fig. 1. The proposed recommendation approach

users and n items. Each row is a users watching history, and each column represents the history of one movie being watched. There are many missing ratings in the matrix. The goal of matrix factorization approach is to predict those missing ratings in the matrix, and then recommend those movies with high predicted ratings to users.



	I_1	I_2	I_3	I_4	···	I_n
U_1	3	4				2
U_2				5		
U_3		1				
U_4			2			3
···						
U_m	2			5		

Fig. 2. An example of rating matrix

Instead of predicting items ratings, in this paper we use the matrix factorization to get items latent vectors. Matrix factorization is based on the assumption that latent vectors of users and items are in the same latent space, and that each user and item can be expressed by a latent vector. Given a rating matrix, matrix factorization decomposes the matrix into the product of users and items latent factors, denoted by $p_u \in \Re^{\kappa}(u = 1, ..., m)$ and $q_i \in \Re^{\kappa}(i = 1, ..., n)$, respectively, where κ is the number of dimensions of latent vectors. The rating of user u on item i is predicted by

$$\widehat{r_{ui}} = p_u q_i^T \tag{1}$$

Latent vectors of users and items are learned by using those known ratings in the matrix. The optimization objective is to minimize the following regularized squared error.

$$\min_{p^*,q^*} \sum_{(u,i)\in D} (r_{ui} - p_u q_i^T)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2)^2 \tag{2}$$

where r_{ui} is the rating of user u on item i; D is the set of all (u, i) pairs for which the rating r_{ui} is known in the matrix. To avoid overfitting and improve the generalization ability, we need to regularize the learned latent factors. Parameter λ is to control the extent of regularization and usually determined by the cross-validation [5]. The latent vectors of all users and items are learned by the gradient descent method.

2.2 Recurrent Neural Network for Latent Vectors Prediction

Neural networks have been applied to recommender systems in recent years [15,17,21]. Literatures [15,16] used a shallow Restricted Boltzmann Machines neural network to predict the ratings. In literatures [17,18], unknown ratings are predicted by auto-encoder neural networks trained by an unsupervised learn algorithm. Paul et al. [19] devised a recommender system which is comprised of two neural networks, one for candidate generation and the other for ranking.

Instead of using a neural network to predict the ratings, this paper uses a recurrent neural network to predict the list of items that interests a user. Recurrent neural networks have been proved to be effective for time series prediction, and as such, we adopt a recurrent neural network to predict the latent vectors of items.

The recurrent network is shown in Fig. 3. Recurrent neural networks all loops with themselves, allowing information to persist. In this process, each step every unit takes previous outputs as one part of inputs. So the process is much like a chain, and naturally fit to model the sequence data.

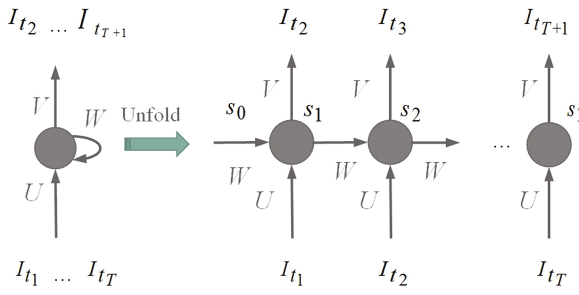


Fig. 3. Predicting latent vectors by RNN.

In Fig. 3, s_t is the state of the RNN at time step t. $U(V)$ is the input(output) weight of the RNN. W is the state weight of the RNN. Inputs of the RNN are items latent vectors. The number of RNN unfolded back steps is denoted by T, which is also the window size of the items sequence.

In this paper, we adopt a popular RNN named Long Short-Term Memory (LSTM). The parameter W is learned by the Backpropagation Through Time algorithm [22], and other parameters are learned by the Backpropagation learning algorithm. We use the following square loss function, which is to be minimized by the learning algorithms.

$$\widehat{r_{ui}} = E_{t_k}(I_{t_k}, \widehat{I_{t_k}}) = \frac{1}{2}(I_{t_k} - \widehat{I_{t_k}})^2 \tag{3}$$

$$E = \sum_{t_k} E_{t_k}(I_{t_k}, \widehat{I_{t_k}}) = \frac{1}{2}\sum_{t_k}(I_{t_k} - \widehat{I_{t_k}})^2 \tag{4}$$

where I_{t_k} is the kth item in the sequence S, and $\widehat{I_{t_k}}$ is the corresponding predicted item. The loss function value is the sum of errors between predicted latent vectors and actual ones. In the item sequence S, each set of $(T+1)$ consecutive items is regarded as a training example. As shown in Fig. 3, latent vectors of the first (last) T items are inputs (outputs) of the RNN.

2.3 Generate a Recommendation List by Predicted Latent Vectors

The output of the RNN is a predicted latent vector. Recall that we have the latent vectors of all items. The n items whose latent vectors are closest to the predicted one are chosen to form the recommendation list.

Euclidean distance is used to calculate the distance of any two latten vectors.

$$dist(q_i, q_j) = \sqrt{\sum_x (q_{ix} - q_j)^2} \tag{5}$$

where q_i and q_j are any two latent vectors.

3 Experiments

To verify the proposed approach, it is applied to a real life dataset, MovieLens (1M), which is a popular dataset to test recommendation approaches. The dataset has total 6040 users and 3952 movies. A user rates a movie after watching it. There are 1000209 ratings, each of which is an integer from 1 to 5. Every user watched at least 20 movies in the dataset.

3.1 Performance Metrics

The commonly used matrix of recall rate is used to evaluate the proposed approach and those comparative approaches. For user i, the recall rate is expressed by

$$recall_i = |recom\_list_i \bigcap target\_list_i| / |target\_list_i| \tag{6}$$

For user i, the last n items in the sequence S are used as test data, and other items are used to train the RNN. As stated in Sect. 2.3, the recommendation list predicted by the RNN, $recom\_list_i$, contains n items. $target\_list_i$ consists of the last n items, that is, the last n movies actually watched by the user.

The average recall rate is defined as follows.

$$avg\_recall = \sum_{i \in Users} recall_i / |Users| \tag{7}$$

where $Users$ represents the set of all users. In the following experiments, the matrix of recall rate refers to the average recall rate.

3.2 Experiment Settings

As stated in Sect. 2, for a user, a movies sequence S can be obtained. In the sequence S, the last 10 movies are chosen as test data, and other movies as training data to train the RNN. In the training data, the size of time window, T, is set as 10. It means that every 10 consecutive movies in the sequence are treated as a training example. The number of items in the recommendation list, s, is given by 10.

The number of dimensions of latent vectors, κ, is set to be 10. The matrix factorization in our approach is realized by the open source software LIBMF. The MF models learning rate is set as 0.05 and the training epoch as 800.

We implement the RNN (LSTM) in the tensorflow framework of version 0.12. The learning rate is given by 0.1 and the training epoch by 6. The parameter of hidden size of the LSTM is set to be 10 and the number of RNN unfolded back steps is set as 9.

3.3 Experiment Results

The proposed approach is compared against the following typical recommendation approaches.

- Item-based KNN [3,4]: item-based k-nearest neighbor collaborative filtering.
- User-based KNN [1,2]: user-based k-nearest neighbor collaborative filtering.
- MF: basic Matrix Factorization recommender.
- Biased MF: biased Matrix Factorization recommender.
- BPMF [9]: Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo.
- PMF [8]: Probabilistic Matrix Factorization.

All above comparative approaches are implemented using the open source platform librec [20]. They use the same training and test data as our approach.

Experiment results are presented in Table 1. Table 1 shows that the proposed approach outperforms all of the matrix factorization approaches and the two neighbor-based CF approaches. Average recall rates obtained by all matrix factorization approaches are smaller than 0.08, and are much smaller than that of

Table 1. Comparison results on MovieLens (1M) dataset.

Methods	Item-based-KNN	User-based-KNN	MF
Recall rate	0.003	0.019	0.004
Methods	BPMF	PMF	Biased MF
Recall rate	0.002	0.008	0.007
Methods	Our approach		
Recall rate	0.024		

Table 2. The produced recommendation list for a user.

Recommendation list of movies	
Movies	Movie types
Regret to Inform(1998)	Documentary
It's a Wonderful Life(1963)	Drama
Charade(1963)	Comedy,Mystery,Romance,Thriller
Toy Story(1995)	Animation,Childrens,Comedy
Run Silent,Run Deep(1958)	War
Beauty and the Beast (1991)	Animation,Childrens,Musical
Day the Earth Stood Still(1951)	Drama,War
Sting(1973)	Comedy,Crime
Great Escape(1963)	Adventure,War
Running Free(2000)	Drama
Actually watched list of movies	
Movies	Movie types
Beauty and the Beast(1991)	Animation,Childrens,Musical
Toy Story(1995)	Animation,Childrens,Comedy
Aladdin(1992)	Animation,Childrens,Comedy
Close Shave(1995)	Animation,Comedy,Thriller
Antz(1998)	Animation,Childrens
Hunchback of Notre Dame(1996)	Animation,Childrens,Musical
Bugs Life(1998)	Animation,Childrens,Comedy
Mulan(1998)	Animation,Childrens
Hercules(1997)	Animation,Childrens,Comedy
Pocahontas(1995)	Animation,Childrens,Musical

our approach. Among all comparative approaches, the user-based KNN obtains the best average recall rate 0.019, which is also lower than that of our approach. For the movie recommendation problem, interests and behaviors of users may change over time and the problem is actually a time series prediction problem.

RNN is a powerful tool for the time series prediction, such that it is more suitable for such problem than matrix factorization approaches, which recommend movies based on predicted ratings and do not consider the effect of time factor on recommendation results.

To observe recommendation results intuitively, Table 2 gives the recommended list produced by our approach for a user. The actually watched movies by the user are also given in the Table. We can see that many movies in the recommendation list belong to the types of animations and comedies. Most of actually watched movies also belong to the two types. It indicates that our approach is able to recommend appropriate movies to the user and those movies reflect the users interests.

4 Conclusions

In this paper, a recommendation approach combining a matrix factorization and a recurrent neural network is proposed. Different from the traditional recommendation approaches based on ratings prediction, this approach considers the items rated by a user as a set of time series data. First, the matrix factorization is used to obtain latent vectors of all items, such that the time series items are transformed into time series latent vectors. Then, the recurrent neural network is taken as a time series prediction mode to predict a latent vector. Finally, a recommendation list is produced by the predicted latent vector. Experiments show that the proposed approach outperforms those comparative approaches and can produce appropriate recommend lists for users.

Acknowledgement. This work was supported in part by National Natural Science Foundation of China (61374204; 61375066)

References

1. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, pp. 43–52. Morgan Kaufmann Publishers Inc., Madison (1998)
2. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work, pp. 175–186. ACM, Chapel Hill (1994)
3. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295. ACM, Hong Kong (2001)
4. Deshpande, M., Karypis, G.: Item-based top-N recommendation algorithms. ACM Trans. Inf. Syst. **22**(1), 143–177 (2004)
5. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)

6. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD Cup and Workshop, pp. 5–8. ACM, California (2007)
7. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM, New York (2008)
8. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: International Conference on Neural Information Processing Systems, pp. 1257–1264. Curran Associates Inc., Vancouver (2007)
9. Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: International Conference on Machine Learning, pp. 880–887. ACM, Helsinki (2008)
10. Liang, D., Altosaar, J., Charlin, L., Blei, D.M.: Factorization meets the item embedding: regularizing matrix factorization with item co-occurrence. In: ACM Conference on Recommender Systems, pp. 59–66. ACM, Boston (2016)
11. Gopalan, P., Charlin, L., Blei, D.M.: Content-based recommendations with poisson factorization. In: International Conference on Neural Information Processing Systems, pp. 3176–3184. MIT Press, Montreal (2014)
12. Mcauley, J., Leskovec, J.: Hidden factors and hidden topics: understanding rating dimensions with review text. In: ACM Conference on Recommender Systems, pp. 165–172. ACM, Hong Kong (2013)
13. Shan, H., Banerjee, A.: Generalized probabilistic matrix factorizations for collaborative filtering. In: IEEE International Conference on Data Mining, pp. 1025–1030. IEEE Computer Society, Sydney (2010)
14. Wang, C., Blei, D.M.: Collaborative topic modeling for recommending scientific articles. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 448–456. ACM, San Diego (2011)
15. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: International Conference on Machine Learning, pp. 791–789. ACM, Corvallis (2007)
16. Georgiev, K., Nakov, P.: A non-IID framework for collaborative filtering with restricted boltzmann machines. In: International Conference on Machine Learning, pp. 1148–1156. Atlanta (2013)
17. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: AutoRec: autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web, pp. 111–112. ACM, Florence (2015)
18. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-N recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp. 153–162. ACM, San Francisco (2016)
19. Covington, P., Jay, A., Emre, S.: Deep neural networks For Youtube recommendations. In: ACM Conference on Recommender Systems, pp. 191–198. ACM, Boston (2016)
20. Guo, G., Zhang, J., Sun, Z., Yorke-Smith, N.: LibRec: a Java library for recommender systems. In: UMAP Workshops (2015)
21. Wu, C., Wang, J., Liu, J., Liu, W.: Recurrent neural network based recommendation for time heterogeneous feedback. Knowl. Based Syst. **109**, 90–103 (2016)
22. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM. In: International Conference on Artificial Neural Networks, pp. 850–855. Edinburgh (1999)