Journal of Systems Engineering and Electronics
Vol. 31, No. 4, August 2020, pp.674 – 684

User space transformation in deep learning based recommendation

WU Caihua[*], MA Jianchao, ZHANG Xiuwei, and XIE Dang

Radar Non-Commissioned Officer School, Air Force Early Warning Academy, Wuhan 430345, China

Abstract: Deep learning based recommendation methods, such as the recurrent neural network based recommendation method (RNNRec) and the gated recurrent unit (GRU) based recommendation method (GRURec), are proposed to solve the problem of time heterogeneous feedback recommendation. These methods out-perform several state-of-the-art methods. However, in RNNRec and GRURec, action vectors and item vectors are shared among users. The different meanings of the same action for different users are not considered. Similarly, different user preference for the same item is also ignored. To address this problem, the models of RNNRec and GRURec are modified in this paper. In the proposed methods, action vectors and item vectors are transformed into the user space for each user firstly, and then the transformed vectors are fed into the original neural networks of RNNRec and GRURec. The transformed action vectors and item vectors represent the user specified meaning of actions and the preference for items, which makes the proposed method obtain more accurate recommendation results. The experimental results on two real-life datasets indicate that the proposed method outperforms RNNRec and GRURec as well as other state-of-the-art approaches in most cases.

Keywords: recommender system, collaborative filtering, time heterogeneous feedback, recurrent neural network, gated recurrent unit (GRU), user space transformation.

DOI: 10.23919/JSEE.2020.000043

1. Introduction

Recently, several recurrent neural network based recommendation methods, such as the recurrent neural network based recommendation method (RNNRec) [1] and the gated recurrent unit (GRU) based neural network recommendation method (GRURec) [2], are proposed to address the problem of time heterogeneous feedback recommendation [1]. In this scenario, different kinds of user feedback with time stamps, such as rating, transaction, browsing, reviewing, sharing and so on, are used to generate personalized recommendation results. It is reported that RNNRec and GRURec generate more accurate recommendation results than several traditional recommendation methods.

In RNNRec and GRURec, actions and items are represented by vectors. Recommendation results are generated according to action vector sequences and the corresponding item vector sequences, which are the input of a recurrent neural network. Action vectors and item vectors are shared among users. That is, the vector of one kind of action or one item remains the same when it is used to generate recommendation for different users. However in practice, the meaning of a same action is different for different users. For example, some users give 3-point to an item because they like it, but for other people, 3-point means not very good. Another example is adding a product into the cart. Some users add a product into the cart because they will buy it later, but some users add a product into the cart only to record it. RNNRec and GRURec do not consider the difference of a same action for different users. Similarly, these methods also ignore the difference of user preference by sharing item vectors among users. To address these problems, in this paper, we modify the models of RNNRec and GRURec. In the proposed model, item vectors and action vectors are transformed into the user space for each user at first. Thus, the transformed action vectors represent the specific meaning for the user, and the transformed item vectors reflect the preference of the user. Then, the transformed action vectors and item vectors are input to the neural networks of RNNRec and GRURec to generate personalized recommendation results.

The main contributions of this paper are twofold. Firstly, we modify the models of RNNRec and GRURec by adding the user space transformation part into the original models, and the modified models are called TransRNNRec and TransGRURec, respectively. In the proposed models, the user specific meaning of the actions and the user preference to the items are considered by transforming the ac-

Manuscript received March 15, 2019.
*Corresponding author.
This work was supported by the National Natural Science Foundation of China (61403350).

tion vectors and item vectors into the user space in the user space transformation part. This makes the proposed models obtain more accurate results than RNNRec and GRURec. Secondly, we verify the proposed models on two real-life datasets. The experimental results indicate that the proposed models outperform other state-of-the-art recommendation methods.

The rest of this paper is organized as follows. In the next section, we briefly review the traditional recommendation methods and the newly proposed deep learning based recommendation methods. Section 3 presents the preliminaries of this paper, including the introduction of the time heterogeneous feedback recommendation problem, RNNRec and GRURec. And then, the proposed TransRNNRec and TransGRURec are introduced in details in Section 4. In Section 5, the proposed methods are compared with some state-of-the-art methods on two large-scale real-life datasets, and the convergence of the proposed methods are also analyzed. Finally, the conclusions and future work are provided in Section 6.

2. Related work

Recently, with the rising of deep learning, several deep learning based recommendation approaches are proposed. In these methods, deep learning models, such as restricted Boltzmann machine (RBM), convolutional neural network (CNN), stacked denoising autoencoders (SDAE), deep structured semantic model (DSSM), recurrent neural network and so on, are used solely or combined. Here, we briefly introduce representative deep learning based recommendation. For more details, please see the surveys [3,4].

Salakhutdinov et al. [5] reported that RBM slightly outperforms singular value decomposition (SVD) on the Netflix dataset for rating prediction. Oord et al. [6] proposed to get music feature vectors from audio data by a CNN for recommendation. Wang et al. [7] combined SDAE [8] and collaborative topic regression (CTR) [9] for recommendation. Elkahky et al. [10] proposed a DSSM based model for cross domain recommendation. Wang et al. [11] proposed a generative adversarial network (GAN) based information retrieval method, which can be used for web search, recommendation and question answering. Recurrent neural network is the most used model in the deep learning based recommendation methods[4]. For example, Zhang et al. [12] designed a recurrent neural network model for ratings prediction. Hidasi et al. [13] proposed a recurrent neural network based approach to predict the next items that the user may click on. Wu et al. [1] proposed RNNRec to solve the problem of time heterogeneous feedback recommendation. Liu et al. [2] added a GRU layer into the model of RNNRec [1] and proposed a GRU based neural network, GRURec, to perform multiple time scales analysis and avoid the gradient vanishing during training. Ebesu et al. [14] proposed a memory networks based recommendation model, called collaborative memory networks (CMN), where neural attention mechanism is also integrated in this model. Li and She [15] proposed to use variational autoencoder (VAE) to learn deep latent representations from content data, and the implicit relationships between items and users are learned from both content and ratings. Liang et al. [16] extended VAE by introducing a different regularization parameter in the objective function for implicit feedback. The parameters in the model are inferred through the Bayesian method.

RNNRec and GRURec are most related to this work. The main difference between these methods and the proposed methods is that item vectors and action items are transformed into the user space for each user in the proposed model. The transformed action vectors and item vectors represent the user specific meaning of the actions and the user preference to the items.

3. Preliminaries

In this section, we first introduce the problem of time heterogeneous feedback recommendation. And then, we briefly introduce two recurrent neural network based recommendation methods, RNNRec [1] and GRURec [2]. The time heterogeneous feedback recommendation problem was first introduced in [1]. In this scenario, the recommender systems try to predict which items the users may prefer in the future according to the user historical feedback with time stamps. For more details, please see [1]. Two recurrent neural network based recommendation models, RNNRec and GRURec, are designed to solve the time heterogeneous feedback recommendation problem. The structures of these models are shown in Fig. 1 and Fig. 2. In these methods, the time heterogeneous feedback recommendation problem is transformed to estimating the probability that a user will prefer an item in the future given the historical feedback with time stamps, $P(j|A_i)$, here $A_i = (a_{i,1}, a_{i,2}, \ldots)$ is the feedback sequence of user i sorted by time stamps. The recommender system will recommend the items that the user may access most likely.

In Fig. 1 and Fig. 2, the current user vector $\boldsymbol{u}$, item vector $\boldsymbol{v}(t)$, feedback vector $\boldsymbol{a}(t)$ are the inputs of the RNNRec model. $\boldsymbol{o}(t)$ is the output of the model. The hidden layer $\boldsymbol{s}(t)$ remembers the state of user historical activities, and this part of the model is recurrent. The hidden layer $\boldsymbol{h}$ represents the relatively stable user preference, and this part of the model is non-recurrent. RNNRec outperforms some state-of-the-art recommendation methods on large-scale real life datasets.

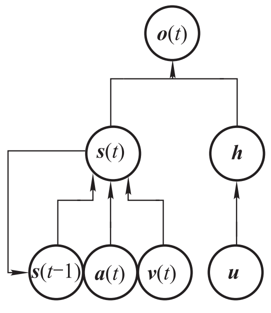


Fig. 1 Structure of RNNRec

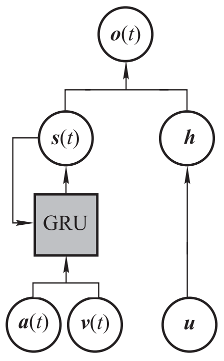


Fig. 2 Structure of GRURec

However, because of the recurrent structure in the model of RNNRec, gradient vanishing may occur during training. And RNNRec is unable to analyze the feedback sequence on multiple time scales, which may lead to some recommendation error. To overcome these two drawbacks, a GRU layer, the grey block in Fig. 2, is added into the model of RNNRec. GRUs are able to prevent gradient vanishing [17]. Furthermore, both long-term and short-term dependencies in sequences can be learned through the reset gates and the update gates in the GRU layer [18]. Thus, the users' historical feedback sequences can be analyzed on multiple time scales. For these reasons, GRURec gets more accuracy results than RNNRec. For more details about these two methods, please see [1] and [2].

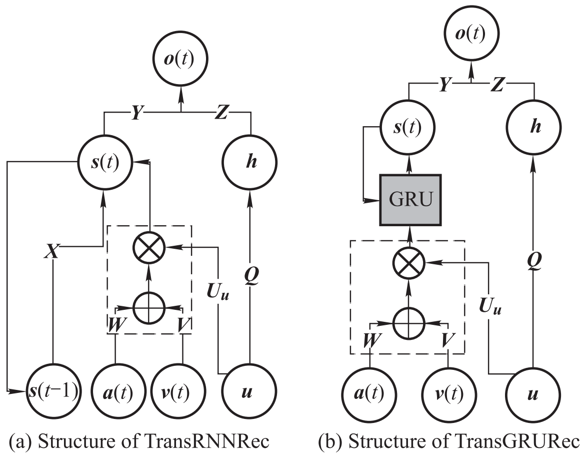
Although RNNRec and GRURec outperform some traditional recommendation methods, the embedding item vectors and action vectors are unchanged among users. It makes the models of RNNRec and GRURec cannot reflect the different meanings of one kind of feedback activity for different users and the different preference of users for a same item. Thus, RNNRec and GURRec may obtain some wrong recommendation results.

4. The proposed method

In this paper, in order to improve the accuracy of RNNRec methods further, we modify the models of RNNRec and GRURec, and propose TransRNNRec and TransGRURec, respectively. In the proposed models, action vectors and item vectors are transformed into the user space for each user at first. The transformed action vectors and item vectors represent the user specific meaning of the actions and user preference to the items, respectively. And then, the transformed action vectors and item vectors are fed into the original models. The space transformation makes the proposed models get more accurate results. Next we will introduce TransRNNRec and TransGRURec in details.

4.1 RNNRec and GRURec with user space transformation

The structures of TransRNNRec and TransGRURec are shown in Fig. 3. The main difference between the proposed models and the original models in Fig. 1 and Fig. 2 is that the item vector $\boldsymbol{v}(t)$ and action vector $\boldsymbol{a}(t)$ in the input layer are transformed into the user space in the proposed model. The parts of user space transforming in the proposed model is marked by dashed box in Fig. 3.



(a) Structure of TransRNNRec (b) Structure of TransGRURec

Fig. 3 Structure of TransRNNRec and TransGRURec

Similar as RNNRec and GRURec, in the proposed model, users, items and actions are represented by hot vectors, which are called the user, item and action vectors, respectively. The current user vector $\boldsymbol{u}$, item vector $\boldsymbol{v}(t)$, action vector $\boldsymbol{a}(t)$ are the inputs of the proposed models. In the original models in Fig. 1, the weight matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ can also be viewed as the item embedding matrix and the action embedding matrix, respectively. Each column of matrix $\boldsymbol{V}$ or $\boldsymbol{W}$ represents the latent feature of an item or an action. The item embedding matrix $\boldsymbol{V}$ and the action embedding matrix $\boldsymbol{W}$ are shared among users in RNNRec and GRURec, and this makes the original models cannot reflect the different meanings of actions for different users and user preference to the items. Thus, in the proposed models, the user space transformation matrix $\boldsymbol{U}_u$ is added for the user $\boldsymbol{u}$, through which the item embedding vector and the action embedding vector are transformed into the user space. And then, the transformed item embedding vector and the action embedding vector are input to the hidden layer. For example, in TransRNNRec, hidden layers are calculated as follows:

$$\boldsymbol{s}(t) = \sigma(\boldsymbol{U}_u(\boldsymbol{V}\boldsymbol{v}(t) + \boldsymbol{W}\boldsymbol{a}(t)) + \boldsymbol{X}\boldsymbol{s}(t-1)) \tag{1}$$

where $\boldsymbol{v}(t)$ and $\boldsymbol{a}(t)$ are the current item vector and the current action vector. $\boldsymbol{s}(t-1)$ is the last hidden layer state. $\boldsymbol{V}$, $\boldsymbol{W}$, $\boldsymbol{X}$ and $\boldsymbol{Q}$ are the weight matrices between the input layer and the hidden layer. Matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ are also called the item embedding matrix and the action embedding matrix, respectively. In (1), $\boldsymbol{U}_u(\boldsymbol{V}\boldsymbol{v}(t) + \boldsymbol{W}\boldsymbol{a}(t))$ is the sum of the transformed item vector and action vector.

In TransGRURec, the reset gate $\boldsymbol{r}$, the update gate $\boldsymbol{z}$ and the candidate hidden layer $\boldsymbol{g}$ in the GRU layer are calculated after the user space transforming, as follows:

$$\boldsymbol{r} = \sigma(\boldsymbol{U}_u(\boldsymbol{W}^r\boldsymbol{a}(t) + \boldsymbol{V}^r\boldsymbol{v}(t)) + \boldsymbol{X}^r\boldsymbol{s}(t-1)) \tag{2}$$

$$\boldsymbol{z} = \sigma(\boldsymbol{U}_u(\boldsymbol{W}^z\boldsymbol{a}(t) + \boldsymbol{V}^z\boldsymbol{v}(t)) + \boldsymbol{X}^z\boldsymbol{s}(t-1)) \tag{3}$$

$$\boldsymbol{g} = \tanh(\boldsymbol{U}_u(\boldsymbol{W}\boldsymbol{a}(t) + \boldsymbol{V}\boldsymbol{v}(t)) + \boldsymbol{X}(\boldsymbol{s}(t-1) \circ \boldsymbol{r})) \tag{4}$$

where $\boldsymbol{W}^r$, $\boldsymbol{V}^r$, $\boldsymbol{X}^r$, $\boldsymbol{W}^z$, $\boldsymbol{V}^z$, $\boldsymbol{X}^z$, $\boldsymbol{W}$, $\boldsymbol{V}$ and $\boldsymbol{X}$ are the weight matrices, and $\circ$ is the element-wise multiplication. In the formulas above $\boldsymbol{U}_u(\boldsymbol{W}^r\boldsymbol{a}(t) + \boldsymbol{V}^r\boldsymbol{v}(t))$, $\boldsymbol{U}_u(\boldsymbol{W}^z\boldsymbol{a}(t) + \boldsymbol{V}^z\boldsymbol{v}(t))$ and $\boldsymbol{U}_u(\boldsymbol{W}\boldsymbol{a}(t) + \boldsymbol{V}\boldsymbol{v}(t))$ are the transforming of the item vector and the action vector.

The calculation of the hidden layer $\boldsymbol{s}(t)$ in TransGRURec is the same with that in GRURec, as follows:

$$\boldsymbol{s}(t) = (1 - \boldsymbol{z}) \circ \boldsymbol{g} + \boldsymbol{z} \circ \boldsymbol{s}(t-1). \tag{5}$$

The hidden layer $\boldsymbol{h}$ and the output $\boldsymbol{o}(t)$ of TransRNNRec and TransGRURec are calculated in the same way of RNNRec and GRURec.

$$\boldsymbol{h} = \sigma(\boldsymbol{Q}\boldsymbol{u}) \tag{6}$$

$$\boldsymbol{o}(t) = \operatorname{soft}\max(\boldsymbol{Y}\boldsymbol{s}(t) + \boldsymbol{Z}\boldsymbol{h}) \tag{7}$$

where $\boldsymbol{Q}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are the weight matrices.

4.2 Model learning

The proposed model can be learned by the back propagation (BP) algorithm or the BP through time (BPTT) algorithm. In this paper, the models of TransRNNRec and TransGRURec are learned through BPTT cooperating with root mean square prop (RMSPROP) [19] algorithm, which is a variation of stochastic gradient descent (SGD). In RMSPROP, different parameters are updated using different learning rates. The parameters with greater gradients in previous training epochs get smaller learning rates, and the parameters with smaller gradients get larger learning rates on the contrary. For more details, please see [19].

4.3 Recommendation generation

The recommendation results are generated in the same way of RNNRec and GRURec. After the models of TransRNNRec and TransGRURec are learned, we calculate the output of the neural networks at the last time stamp for each user. And then, pick up the K largest elements of the output. The indexes of these elements are the IDs of the items recommended to this user.

4.4 Computational complexity

In TransRNNRec, the time complexity of computing the model output is about $o((D+n)D)$, here D is the dimension of the hidden layer, and n is the number of items. The complexity of updating weight matrices between the output layer and the hidden layer, $\boldsymbol{Y}$ and $\boldsymbol{Z}$, once is about $o(nD)$. The complexity of updating matrices $\boldsymbol{X}$ and $\boldsymbol{U}_u$ once is about $o(D^2)$. Only one column of matrice $\boldsymbol{W}$, $\boldsymbol{V}$ or $\boldsymbol{Q}$ is updated for a single training sample, so the complexity of updating these matrices once is about $o(D)$. If these matrices are updated by BPTT, the complexities are about $o(TD^2)$ and $o(TD)$, here T is the number of the unfolded time steps. In recommender systems, $n \gg D$, so the complexity of processing a single training sample through the BP algorithm is about $o(TD)$. If the BPTT algorithm is used to train the model, the complexity of processing a single training sample is slightly higher than $o(TD)$. If there are n_s training samples, the complexity of training the model in one iteration through the BPTT algorithm is slightly higher than $o(n_s nD)$.

Similarly, in TransGRURec, the complexity of computing the output is about $o((D+n)D)$. The complexity of training the TransGRURec model in one iteration through the BPTT algorithm is slightly higher than $o(n_s nD)$.

Because there are more weight matrices in TransGRURec, the complexity of TransGRURec is higher than that of TransRNNRec. Because of the user space transforming in models of TransRNNRec and TransGRURec, the complexities of TransRNNRec and TransGRURec are slightly higher than those of RNNRec and GRURec, respectively.

5. Experimental results and discussions

In this section, the proposed methods are verified on two large-scale real-life datasets. First, the datasets and the quality metrics are introduced. Then, the experiment setting is presented. Next, the proposed methods are compared with several state-of-the-art approaches. Finally, convergence analysis of the proposed method is provided.

5.1 Datasets and metrics

We evaluate the proposed methods, TransRNNRec and

TransGRURec, on the Taobao 2014 dataset and the Lastfm dataset [20]. Table 1 summarizes the statistics of these datasets. A total of 884 users' 182 880 activities on 9 531 items on Taobao (www.taobao.com) are collected in Taobao 2014 dataset. There are four kinds of activities in this dataset, including clicking, adding into the favorites list, adding into the cart and buying.

Table 1 Statics of the datasets

Data set	Taobao 2014	Lastfm
Number of users	884	1 892
Number of items	9 531	17 632
Number of feedback	182 880	186 479
Average feedback per user	206.88	98.57
Average feedback per item	19.18	10.58
Density	2.17×10^{-2}	5.59×10^{-3}
Kinds of feedback	4	1

There are 1 892 users and 17 632 artists in the Lastfm (http://www.lastfm.com) dataset [17]. A total of 186 479 tags assigned by users are also recorded. All of these tag activities are transformed into one kind of feedback in the experiments to verify the proposed methods in the circumstance of implicit feedback.

The recommendation results are evaluated by F1 score for each user, which is often used in information retrieval and classification to measure the performance. F1 score is the weighted average of precision and recall, as follows:

$$\mathrm{F1} = \frac{2 \cdot \mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{8}$$

$$\mathrm{precision} = \frac{|\mathrm{prediction\_set} \cap \mathrm{reference\_set}|}{|\mathrm{prediction\_set}|} \tag{9}$$

$$\mathrm{recall} = \frac{|\mathrm{prediction\_set} \cap \mathrm{reference\_set}|}{|\mathrm{reference\_set}|} \tag{10}$$

where $\mathrm{prediction\_set}$ is the item set recommended to the users obtained by the recommendation method. $\mathrm{reference\_set}$ is the relevant item set containing all of the items in the testing data. $|\cdot|$ is the number of elements in the set.

5.2 Experiment settings

In the time heterogeneous feedback recommendation problem, recommendation results are generated according to the historical feedback activities, and verified by comparing them with the feedback activities in the future. Therefore, the datasets are divided into the training data and the testing data according to the time stamps of the feedback firstly. For the Lastfm dataset, we pick the last 10 or 20 tag activities of each user as the testing data, and the rest are treated as the training data. For the Taobao 2014 dataset, we select activities in the last 10 or 20 days as the testing data, and the rest are treated as the training data. In the experiments, the items in the testing data are set as the relevant items ($\mathrm{reference\_set}$ in (9) and (10)). Then, we train TransRNNRec, TransGRURec and the compared method. After that, we generate the recommendation item lists ($\mathrm{prediction\_set}$ in (9) and (10)) using the trained methods. Finally, F1 scores of the recommendation results obtained by each method are calculated.

The proposed methods are verified in the cases of $D = 16$ or $D = 32$ in the experiments. Here, D is the dimension of the hidden layer.

The learning rate of the proposed methods is selected according to the experimental results in Section 5.4. The best learning rate of TransRNNRec and TransGRURec on the Taobao 2014 dataset is 0.001. The best learning rates of TransRNNRec and TransGRURec on the Lastfm dataset are 0.01 and 0.001, respectively. For convenience, the learning rate of the proposed methods is set to 0.001 on the Taobao 2014 dataset and 0.01 on the Lastfm dataset.

According to [19], the decay rate of RMSPROP is set to 0.9 or 0.95 typically. Thus, the decay rate is set to 0.95 on all datasets.

The proposed methods are implemented in Python using Theano and accelerated by GPUs.

5.3 Comparison

The proposed methods are compared with four state-of-the-art recommendation methods, weighted regularized matrix factorization (WRMF) [21], Bayesian personalized ranking for matrix factorization (BPRMF) [22], weighted BPRMF [23] and SoftMarginRankingMF [24] on the Taobao 2014 dataset and the Lastfm dataset [20]. Three classical collective filtering methods, the user-based k-nearest neighbors recommendation method (KNN (user)), the item-based k-nearest neighbors recommendation method (KNN(item)) and the most popular item recommendation method (Most Popular) are taken as baselines in the experiments. The proposed method is also compared with RNNRec and GRURec to illustrate the improvement of the user space transforming. The compared methods are briefly introduced as follows:

(i) KNN (user), the user-based k-nearest neighbors recommendation method;

(ii) KNN (item), the item-based k-nearest neighbors recommendation method;

(iii) Most Popular, which recommends the most popular items to the users;

(iv) WRMF [21], where the weighted sum-of-square of matrix factorization errors is minimized;

(v) BPRMF [22], where the objective function is pairwise comparison error;

(vi) Weighted BPRMF [23], an extension of BPRMF, where a weight is given to each pair of comparison;

(vii) SoftMarginRankingMF [24], an extension of maximum margin matrix factorization (MMMF) [25], where the objective function is the ordinal regression score;

(viii) RNNRec, a recurrent neural network based recommendation method without user space transforming;

(ix) GRURec, a GRU based recommendation method without user space transforming.

All of the compared methods except RNNRec and GRURec are implemented in MyMediaLite [26]. In the experiments, the parameters of the compared methods are set to the optimal values stated in the original literature. The parameters of RNNRec and GRURec are the same as those of TransRNNRec and TransGRURec. We train RNNRec, GRURec and the proposed methods through the BPTT algorithm with RMSPROP. The unfolded steps of BPTT is set to 15 for these methods. The experiment is performed five times for each case. Mean and standard deviations (in the brackets) of F1@10 and F1@20 are calculated and listed in Tables 2 – 5 to make the comparison clear. The best results with statistical significance are shown in bold. In the tables, Test=10 means that the last 10 feedback activities of each user in the Lastfm dataset are selected as the testing data, or the feedback activities in last 10 days of each user in the Taobao 2014 dataset are selected as the testing data. The meaning of Test=20 is similar to that of Test=10.

Table 2 Comparison results on Taobao 2014 Dataset (Test=10)

Method	$D = 16$		$D = 32$	
	F1@10	F1@20	F1@10	F1@20
KNN(user)	0.026 0	0.025 8	0.026 0	0.025 8
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
KNN(item)	0.021 4	0.022 3	0.021 4	0.022 3
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
Most popular	0.024 8	0.025 7	0.024 8	0.025 7
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
WRMF	0.030 8	0.029 5	0.029 8	0.028 0)
	(0.000 5)	(0.000 4)	(0.000 7)	(0.000 4
BPRMF	0.017 1	0.018 2	0.017 5	0.008 4
	(0.001 3)	(0.001 3)	(0.001 5)	(0.001 0)
Weighted BPRMF	0.007 7	0.007 8	0.007 6	0.008 4
	(0.001 3)	(0.000 6)	(0.001 4)	(0.000 9)
SoftmarginRankingMF	0.009 8	0.011 6	0.011 1	0.011 7
	(0.001 8)	(0.001 6)	(0.001 7)	(0.000 9)
RNNRec	0.048 1	0.048 2	0.045 8	0.046 6
	(0.002 7)	(0.002 2)	(0.002 2)	(0.002 0)
TransRNNRec	0.062 3	**0.058 3**	**0.066 9**	**0.059 4**
	(0.003 3)	**(0.001 5)**	**(0.001 7)**	**(0.001 5)**
GRURec	0.051 4	0.048 8	0.051 9	0.049 7
	(0.000 8)	(0.000 9)	(0.002 0)	(0.001 1)
TransGRURec	**0.063 3**	0.058 0	0.065 8	0.059 3
	(0.001 8)	(0.002 0)	(0.001 9)	(0.001 3)

On the Taobao 2014 dataset, as shown in Table 2 and Table 3, the proposed methods outperform the original methods in all cases. Compared with RNNRec and GRURec, TransRNNRec and TransGRURec improve F1 scores by about 20% on average, respectively. The main reason is that there is no user space transformation part in the models of RNNRec and GRURec, and the user action and item information are transformed into user space in the proposed methods, which can improve the recommendation accuracy.

Table 3 Comparison results on Taobao 2014 Dataset (Test=20)

Method	$D = 16$		$D = 32$	
	F1@10	F1@20	F1@10	F1@20
KNN(user)	0.036 5	0.038 9	0.036 5	0.038 9
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
KNN(item)	0.028 3	0.032 1	0.028 3	0.032 1
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
Most popular	0.032 6	0.038 0	0.032 6	0.038 0
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
WRMF	0.039 3	0.044 3	0.035 4	0.041 0
	(0.000 9)	(0.000 3)	(0.001 2)	(0.000 9)
BPRMF	0.022 1	0.027 1	0.025 8	0.029 6
	(0.001 3)	(0.001 2)	(0.001 2)	(0.001 4)
Weighted BPRMF	0.009 0	0.011 2	0.010 4	0.029 6
	(0.000 4)	(0.000 6)	(0.001 1)	(0.001 4)
SoftmarginRankingMF	0.013 7	0.017 9	0.014 6	0.018 5
	(0.001 5)	(0.001 7)	(0.002 7)	(0.002 0)
RNNRec	0.057 6	0.064 1	0.057 3	0.063 7
	(0.002 1)	(0.001 6)	(0.002 6)	(0.002 1)
TransRNNRec	0.072 2	0.074 5	**0.076 4**	**0.076 8**
	(0.000 5)	(0.001 5)	**(0.001 3)**	**(0.000 7)**
GRURec	0.061 0	0.068 0	0.062 6	0.068 0
	(0.001 2)	(0.000 9)	(0.001 8)	(0.001 8)
TransGRURec	**0.073 2**	**0.075 8**	0.074 4	0.075 7
	(0.001 0)	**(0.001 0)**	(0.001 7)	(0.001 9)

Table 4 Comparison results on Lastfm dataset (Test=10)

Method	$D = 16$		$D = 32$	
	F1@10	F1@20	F1@10	F1@20
KNN(user)	0.023 1	0.021 4	0.023 1	0.021 4
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
KNN(item)	0.025 5	0.024 5	0.025 5	0.024 5
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
Most popular	0.013 9	0.011 9	0.013 9	0.011 9
	(0.000 0)	(0.000 0)	(0.000 0)	(0.000 0)
WRMF	0.023 1	0.023 1	0.023 3	0.023 0
	(0.000 2)	(0.000 2)	(0.000 2)	(0.000 5)
BPRMF	0.017 5	0.017 1	0.018 5	0.017 5
	(0.001 3)	(0.000 6)	(0.000 7)	(0.000 4)
Weighted BPRMF	0.017 7	0.017 7	0.018 8	0.018 8
	(0.001 1)	(0.000 7)	(0.001 4)	(0.001 4)
SoftmarginRankingMF	0.010 6	0.011 3	0.009 7	0.010 4
	(0.001 7)	(0.001 5)	(0.001 1)	(0.001 1)
RNNRec	0.055 0	0.037 4	0.070 5	0.047 8
	(0.001 5)	(0.001 1)	(0.001 5)	(0.001 3)
TransRNNRec	0.077 9	0.050 6	0.094 5	0.057 0
	(0.003 5)	(0.001 3)	(0.001 5)	(0.001 2)
GRURec	**0.079 8**	**0.055 8**	**0.111 6**	**0.074 7**
	(0.002 5)	**(0.000 7)**	**(0.001 6)**	**(0.001 7)**
TransGRURec	0.054 0	0.040 3	0.089 0	0.056 0
	(0.003 4)	(0.001 9)	(0.005 7)	(0.002 6)

On the Lastfm dataset, TransRNNRec obtains more accuracy results than RNNRec in all cases. The F1 scores obtained by TransRNNRec are about 30% higher than those obtained by RNNRec on average. The reason is the same as that on Taobao 2014 dataset. The user action and item information are transformed into user space in the user space transformation part in TransRNNRec, and this improves the recommendation accuracy. However, TransGRURec does not outperform GRURec on the Lastfm dataset. The possible reason is that there are only one kind of feedback in the Lastfm dataset. The advantage of user space transforming may not be obvious in this situation. Furthermore, the Lastfm dataset is sparser than the Taobao 2014 dataset. The model of TransGRURec is more complex than that of GRURec. More training data are needed for TransGRURec to get better results.

Table 5 Comparison results on Lastfm dataset (Test=20)

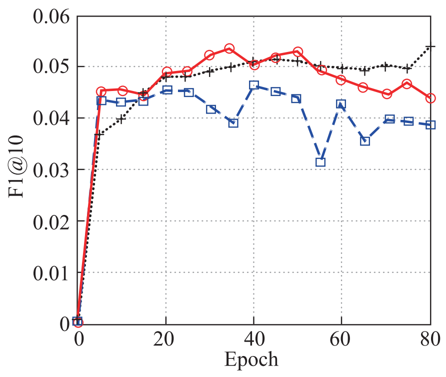
Method	$D=16$		$D=32$	
	F1@10	F1@20	F1@10	F1@20
KNN(user)	0.030 9 (0.000 0)	0.033 1 (0.000 0)	0.030 9 (0.000 0)	0.033 1 (0.000 0)
KNN(item)	0.032 8 (0.000 0)	0.035 7 (0.000 0)	0.032 8 (0.000 0)	0.035 7 (0.000 0)
Most popular	0.016 7 (0.000 0)	0.018 5 (0.000 0)	0.016 7 (0.000 0)	0.018 5 (0.000 0)
WRMF	0.033 9 (0.000 3)	0.036 9 (0.000 3)	0.034 6 (0.000 4)	0.037 8 (0.000 3)
BPRMF	0.024 8 (0.001 3)	0.027 2 (0.001 1)	0.027 6 (0.000 7)	0.029 9 (0.001 3)
Weighted BPRMF	0.024 8 (0.000 9)	0.027 2 (0.001 1)	0.027 6 (0.000 7)	0.029 9 (0.001 3)
SoftmarginRankingMF	0.014 4 (0.001 3)	0.017 1 (0.001 4)	0.013 4 (0.001 8)	0.017 5 (0.000 9)
RNNRec	0.056 7 (0.000 8)	0.043 7 (0.000 9)	0.078 0 (0.002 2)	0.059 0 (0.002 7)
TransRNNRec	0.082 5 (0.001 5)	0.056 8 (0.000 4)	0.095 7 (0.001 3)	0.062 8 (0.000 7)
GRURec	**0.095 7 (0.002 9)**	**0.071 6 (0.002 4)**	**0.114 2 (0.001 4)**	**0.085 3 (0.000 7)**
TransGRURec	0.051 4 (0.003 6)	0.041 9 (0.002 0)	0.076 8 (0.005 1)	0.054 7 (0.003 5)

In all cases, not only RNNRec and GRURec but also the proposed methods outperform other compared methods. This indicates the recurrent neural network based models are more suitable to deal with the time heterogeneous feedback recommendation problem than the traditional methods, including the matrix factorization based methods, such as WRMF, BPRMF, weighted BPRMF and SoftMarginRankingMF.

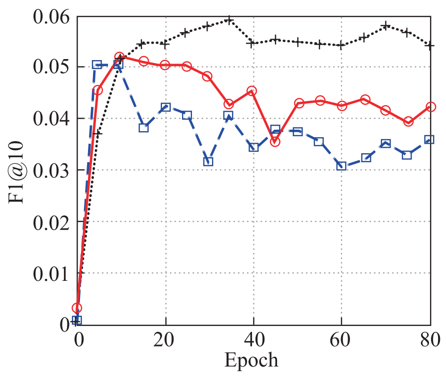
The results obtained by the proposed methods in the cases of $D = 32$ is better than those in the cases of $D = 16$. This indicates that higher hidden layer dimensions can improve the accuracy of the recommendation results of the proposed method. However, training the model with higher hidden layer dimensions needs more computing time. A suitable hidden layer dimension should be selected by trading off between accuracy and training time according to the application scenario.

5.4 Convergence analysis

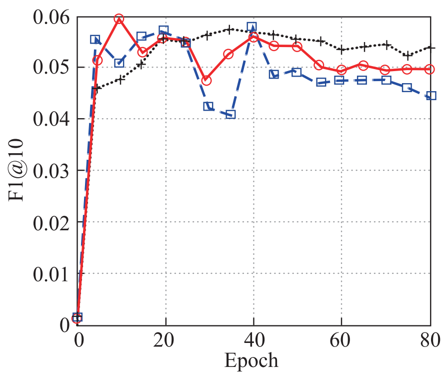
To demonstrate the convergence of the proposed methods, we draw the curves of the testing F1@10 and F1@20 during the training process at different learning rates in Fig. 4 and Fig. 5. The learning rates are set to 0.01, 0.005 and 0.001 on the Taobao 2014 dataset and the Lastfm dataset.



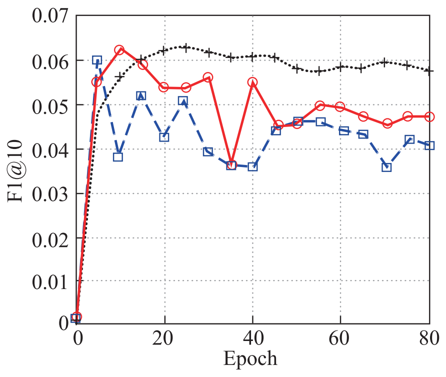
(a) TransRNNRec, Test=10, *D*=16



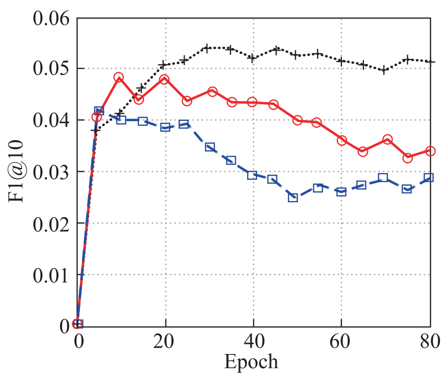
(b) TransRNNRec, Test=10, *D*=32



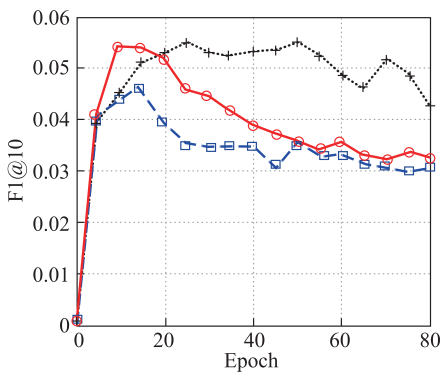
(c) TransRNNRec, Test=20, *D*=16



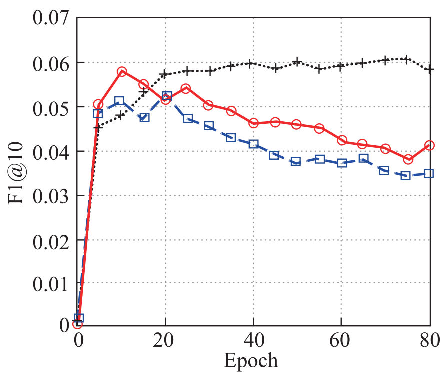
(d) TransRNNRec, Test=20, *D*=32



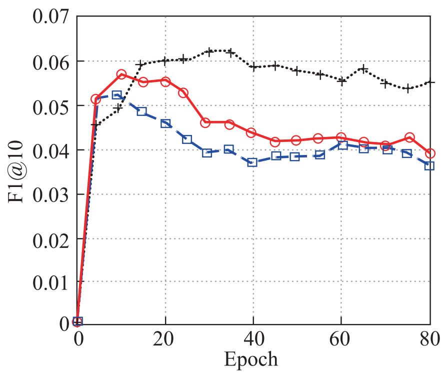
(e) TransGRURec, Test=10, *D*=16



(f) TransGRURec, Test=10, *D*=32



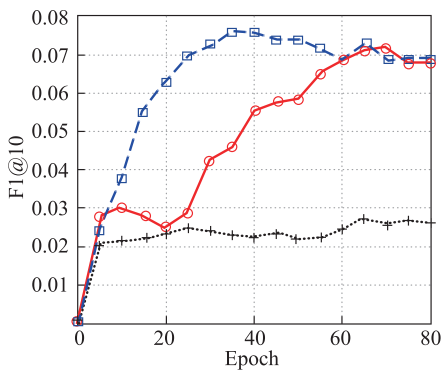
(g) TransGRURec, Test=20, *D*=16



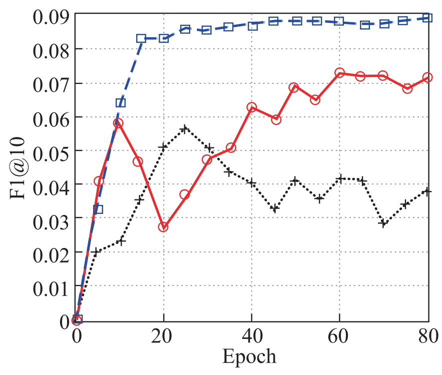
(h) TransGRURec, Test=20, *D*=32

--□-- : Learn rate=0.01; —○— : Learn rate=0.005;
···+··· : Learn rate=0.001.

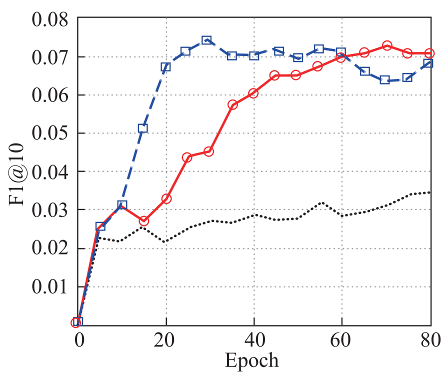
Fig. 4 Testing F1 scores on Taobao 2014 dataset during the training of TransRNNRec and TransGRURec



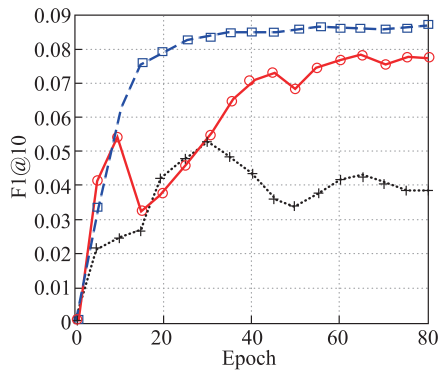
(a) TransRNNRec, Test=10, *D*=16



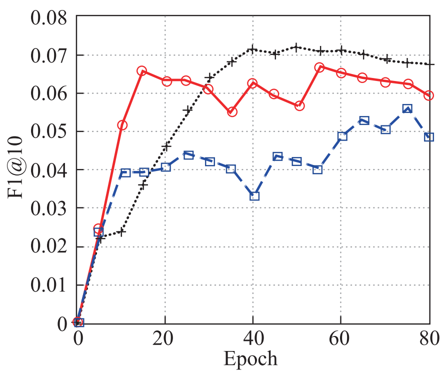
(b) TransRNNRec, Test=10, *D*=32



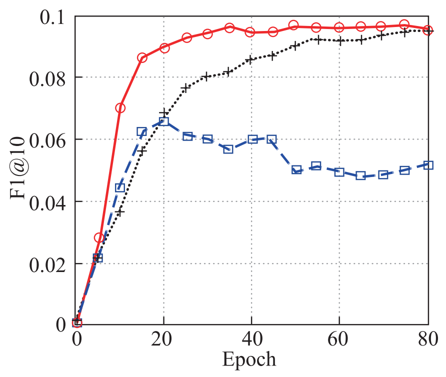
(c) TransRNNRec, Test=20, *D*=16



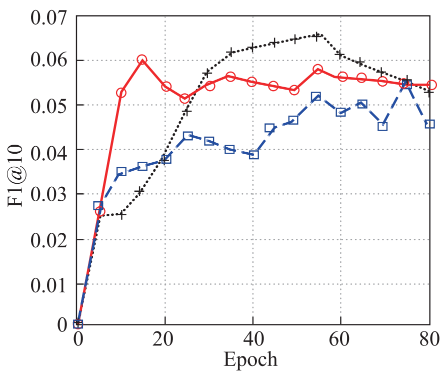
(d) TransRNNRec, Test=20, *D*=32



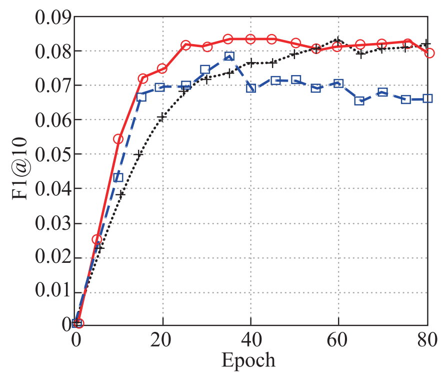
(e) TransGRURec, Test=10, *D*=16



(f) TransGRURec, Test=10, *D*=32



(g) TransGRURec, Test=20, *D*=16



(h) TransGRURec, Test=20, *D*=32

–⊡–: Learn rate=0.01; –○–: Learn rate=0.005;
···+···: Learn rate=0.001.

Fig. 5 Testing F1 scores on Lastfm dataset during the training of TransRNNRec and TransGRURec

From Fig. 4(a) – Fig. 4(d) and Fig. 5(a) – Fig. 5(d), it can be observed that the converge speed of TransRNNRec at a lower learning rate is slower than that at a higher learning rate. On the Taobao 2014 dataset, the training progress of TransRNNRec converges within about 20 epoches. The best result is obtained when the learning rate is set to 0.001 on the Taobao 2014 dataset. When the learning rate is high (0.01 and 0.005), the curves of F1 score fluctuate up and down in a narrow range after several epoches. And in case of lower learning rates, the curves of F1 score are relatively smooth after convergence. This indicates that the lower learning rate can avoid fluctuation during training. On the Lastfm dataset, the best learning rate of TransRNNRec is 0.01, and the training of TransRNNRec converges within 40 epoches at the learning rate of 0.01.

According to the curves in the Fig. 4(e) – Fig. 4(h) and Fig. 5(e) – Fig. 5(h), the best learning rate of TransGRURec on the Taobao 2014 dataset is 0.001. TransGRURec converges within 20 epoches on the Taobao 2014 dataset. The converging process of TransGRURec on the Taobao 2014 dataset is very similar to that of TransRNNRec on this dataset. The best learning rate of TransGRURec on the Lastfm dataset is 0.001. The converging speeds of TransGRURec at lower learning rates (0.005 and 0.001) on this dataset are very slow. Especially at the learning rate of 0.001, more than 80 epoches are needed for convergence. The reason is that the Lastfm dataset is sparser than the Taobao 2014 dataset. Using more training samples can speed up the convergence.

6. Conclusions and future work

In this paper, we modify the model of RNNRec and GRURec, and propose TransRNNRec and TransGRURec, in which the action vectors and the item vectors are transformed into user space before entering the recurrent neural network. This makes the transformed action vectors and item vectors can represent user specified action meaning and the preference to items. This advantage makes the proposed methods outperform the original methods and other state-of-the-art recommendation methods.

It is reported that the external knowledge can help to improve the recommendation accuracy. For example, Oramaset al. [27] linked tags and textual descriptions in a music recommender system to the external knowledge graph, and obtained graph based item feature for recommendation. One of the future research direction is to combine the information in the recommender system and the external knowledge to generate more reasonable recommendation. External knowledge can be used to generate recommendation reasons, and it is also a valuable research direction.

Another possible research direction is to improve the training sample utilization efficiency of recurrent neural network based recommendation. The recurrent neural network based recommendation models are more complex than the traditional recommendation models, and more training samples are needed to get satisfied results for these models. Improving the training sample utilization efficiency of these models can make them more widely applied.

References

[1] WU C H, WANG J W, LIU J T, et al. Recurrent neural network based recommendation for time heterogeneous feedback. Knowledge-Based Systems, 2016, 109: 90 – 103.

[2] LIU J T, WU C H, WANG J W. Gated recurrent units based neural network for time heterogeneous feedback recommendation. Information Sciences, 2018, 423: 50 – 65.

[3] LIU J T, WU C H. Deep learning based recommendation: a survey. Proc. of the International Conference on Information Science and Applications, 2017: 451 – 458.

[4] ZHANG S, YAO L, SUN A. Deep learning based recommender system: a survey and new perspectives. http://arxiv.org/abs/1707.07435.

[5] SALAKHUTDINOV R, MNIH A, REYHINTON G. Restricted Boltzmann machines for collaborative filtering. Proc. of the 24th International Conference on Machine Learning, 2007: 791 – 798.

[6] OORDA V D, DIELEMAN S, SCHRAUWEN B. Deep content-based music recommendation. Proc. of the Annual Conference on Neural Information Processing Systems, 2013: 2643 – 2651.

[7] WANG H, WANG N, YEUNG D Y. Collaborative deep learning for recommender systems. Proc. of the 21st International Conference on Knowledge Discovery and Data Mining, 2015: 1235 – 1244.

[8] VINCENT P, LAROCHELLE H, LAJOIE I, et al. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 2010, 11(12): 3371 – 3408.

[9] WANG C, BLEI D M. Collaborative topic modeling for recommending scientific articles. Proc. of the 17th International Conference on Knowledge Discovery and Data Mining, 2011: 448 – 456.

[10] ELKAHKY A M, SONG Y, HE X. A multi-view deep learning approach for cross domain user modeling in recommendation systems. Proc. of the 24th International Conference on World Wide Web, 2015: 278 – 288.

[11] WANG J, YU L, ZHANG W, et al. Irgan: a minimax game for unifying generative and discriminative information retrieval models. Proc. of the 40th International Conference on Research and Development in Information Retrieval, 2017: 515 – 524.

[12] ZHANG J, CAI H, HUANG T, et al. Distributional representation model for collaborative filtering. http://arxiv.org/abs/1502.04163.

[13] HIDASI B, KARATZOGLOU A, BALTRUNAS L, et al. Session-based recommendations with recurrent neural networks. http://arxiv.org/abs/1511.06939.

[14] EBESU T, SHEN B, FANG Y. Collaborative memory network for recommendation systems. Proc. of the Special Interest Group on Information Retrieval, 2018: 515 – 524.

[15] LI X P, SHE J. Collaborative variational autoencoder for recommender systems. Proc. of the 23th ACM SIGKDD Conference Knowledge Discovery and Data Mining, 2017: 305 – 314.

[16] LIANG D, KRISHNAN R G, HOFFMANM D, et al. Variational autoencoders for collaborative filtering. Proc. of the 27th World Wide Web Conference, 2018: 689 – 698.

[17] CHUNG J, GULCEHRE C, CHO K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. http://arxiv.org/abs/1412.3555.

[18] CHO K, MERRIENBOER B V, GULCEHRE C, et al. Learning phrase representations using RNN encoder decoder for statistical machine translation. http://arxiv.org/abs/1406.1078.

[19] HINTON G, SRIVASTAVA N, SWERSKY K. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning. http://www.cs.toronto.edu/∼tijmen/csc321/slides/lecture_slides_lec6.pdf.

[20] CANTADOR I, BRUSILOVSKY P, KUFLIK T. Second workshop on information heterogeneity and fusion in recommender systems. Proc. of the 5th ACM Conference on Recommender Systems, 2011: 387 – 388.

[21] HU Y, KOREN Y, VOLINSKY C. Collaborative filtering for implicit feedback datasets. Proc. of the 8th IEEE International Conference on Data Mining, 2008: 263 – 272.

[22] GANTNER Z, DRUMOND L, FREUDENTHALER C, et al. Bayesian personalized ranking for non-uniformly sampled items. Proc. of the Knowledge Discovery and Data Mining Cup and Workshop, 2011: 231 – 247.

[23] RENDLE S, FREUDENTHALER C, GANTNER Z, et al. BPR: Bayesian personalized ranking from implicit feedback. Proc. of the 25th Conference on Uncertainty in Artificial Intelligence, 2009: 452 – 461.

[24] WEIMER M, KARATZOHLOU A, SMOLA A. Improving maximum-margin matrix factorization. Machine Learning, 2008, 72(3): 263 – 276.

[25] SREBRO N, RENNIE J D M, JAAKKOLA T S. Maximum-margin matrix factorization. Proc. of the Annual Conference on Neural Information Processing Systems, 2005: 1329 – 1336.

[26] GANTNER Z, RENDLE S, FREUDENTHALER C, et al. Mymedialite: a free recommender system library. Proc. of the 5th ACM Conference on Recommender Systems, 2011: 305 – 308.

[27] ORAMAS S, OSTUNI V C, NOIAT D, et al. Sound and music recommendation with knowledge graphs. ACM Trans. on Intelligent Systems and Technology, 2016, 8(2): 1 – 21.

Biographies

WU Caihua was born in 1980. She received her B.S. and M.S. degrees in computer application from Ordnance Engineering College in 2003 and 2006 respectively, and Ph.D. degree in weapon system and application from Ordnance Engineering College in 2009. She is currently a lecturer in Air Force Early Warning Academy. Her research interests include machine learning, software reliability and information system.

E-mail: wucaihua2009@163.com

MA Jianchao was born in 1972. He received his B.S. degree in command automation engineering from Air Force Radar Academy in 1994, and M.S. degree in military intelligence from Air Force Radar Academy in 1997. He is currently an associate professor in Air Force Early Warning Academy. His research interests include machine learning and information system.

E-mail: 549464509@qq.com

ZHANG Xiuwei was born in 1980. He received his B.S. degree in computer science and technology from Air Force Radar Academy in 2001, and M.S. degree in computer application from Wuhan University of Technology in 2009. He received his Ph.D. degree in computer software and theory from Wuhan University in 2014. He is currently a lecturer in Air Force Early Warning Academy. His research interests include machine learning and information system.
E-mail: 39353745@qq.com

XIE Dang was born in 1979. He received his B.S. degree in information and computing science from Hangzhou University of Electronic Science and Technology in 2003, and M.S. degree in information and signal processing from Air Force Radar Academy in 2009. He is currently a lecturer in Air Force Early Warning Academy. His research interests include machine learning and information system.
E-mail: 9972653@qq.com