SR-RNN: Short-Term Recommendation With Recurrent Neural Networks

Guang Li
Harbin Engineering University
liguang@hrbeu.edu.cn

Yan Chu*
Harbin Engineering University
chuyan@hrbeu.edu.cn

Xuemeng Song
Shandong University
sxmustc@gmail.com

Chenqi Shan
Harbin Engineering University
shanchenqi@hrbeu.edu.cn

***Abstract*—Collaborative filtering is a popular recommendation algorithm that bases its predictions and recommendations on the ratings or behaviors of other users in the system. However, it needs to use all the ratings or behaviors of users rather than the users' recent ratings or behaviors to predict the items. Therefore, it might ignore the consumers' habits changing with time. In the short-term recommendation system we proposed, one item that users most likely to consume is recommended based on their recent ratings or behaviors. In this paper, we build a recurrent neural network to address the problem concerning on a time sequence and use gated recurrent units in recurrent neural network. The network treats a user's recent ratings or behaviors as a sequence, and each hidden layer models a user's rating or behavior which is in order. Furthermore, we integrate the gated recurrent unit with back propagation neural network to increase the prediction accuracy. And our method get higher precision accuracy in experiment.**

***Index Terms*—Collaborative filtering, recurrent neural network, recommendation system, gated recurrent units, back propagation neural network.**

I. Introduction

Collaborative filtering takes advantage of information from a group of people with similar interests to recommend. Its assumption is that if two users' preference history is similar in past, they are likely to consume the same set of items. In general, collaborative filtering can be divided into two classes, memory-based and model-based algorithms. Memory-based algorithm utilizes the entire user-item database to generate a prediction. And model-based collaborative filtering algorithms provide item recommendation by first developing a model of user ratings. Algorithms in this category take a probabilistic approach and envision the collaborative filtering process as computing the expected value of a user prediction, given his/her ratings on other items. The model building process is performed by different machine learning algorithms such as Bayesian network, clustering, and rule-based approaches.[1], [2], [3], [4] In video-sharing website like Netflix, the website would recommend videos for users based on not only the user's watching behavior, but also that of a group of similar-minded users. Matrix factorization is a main method for collaborative filtering.[5] MF techniques are usually effective and allow us to discover the latent features between items and users. Its principle is simple, and easy to program. Although great success has been achived by existing collaborative filtering methods, few of them take the temporal changes of user interests into consideration. Taking a user's watched list of movies as an example, initially, user u_i likes to watch action films, and later, he/she may prefer to watch science fiction movie. On the other hand, user u_j may first like to watch science fiction movie but change to action films later. In MF based recommendation system, if they give the film a similar rating, system will recommend them the same movies. However, it is obvious that user u_i will likely to watch science fiction movie, and user u_j will likely to watch action films.

Towards this end, we build a prediction model based on recurrent neural networks. RNNs, which has been proven to be effective to sequence modeling[6], to take the temporal factor of users' interests into consideration for recommendation. One of the advantages of RNNs is the idea that they might be able to connect previous information to the present task. Therefore, it has a strong power to solve sequence prediction problems. In recommendation system, a user's behavior is which item he/she consumed at time t, it's a one-hot encoding vector, for example there are four videos in website, if user u_i watched the third video at time t, his/her behavior is [0,0,1,0] at that time. We consider each user's behavior as a unit in the RNNs. Then, we can use these sequence information to predict the next unit's state. However, sometimes we only need to look at recent behavior to predict, for example if user u_i watched $DieHard1$ at time t_1, he/she may likes to watch $DieHard2$ at time t_2. So, in that case, we do not need to consider the user's behavior before t_1. Gated recurrent unit (GRU) is powerful tool for that problem. In addition, we also build a back propagation neural network (BPNN) , the input of BPNN is the vector of each user's behaviors form time t_1 to time t_2 and calculate the probability of video which he/she will watch. At last, we combine GRU with BPNN to produce final results.

In this paper we argue that RNNs can be applied to short-term recommendation, and we also combine the GRU with the BPNN. The rest of the paper is organized as following. Section 2 will introduce the related work about MF based recommendation, back propagation neural network (BPNN) , basic RNNs and GRU. Section 3 will introduce short-term recommendation, the short-term recommendation with RNN and how to combine the GRU and BP model. We also will introduce the methods of MF, BPNN and RNNs training. Section 4 will discuss the experimental study and Section 5 will conclude this paper and point out the future work.

II. Related Work

A. MF based recommendation

In a recommendation system such as MovieLens [7], there are a group of users and a set of movies. Each user has rated some movies that he/she has watched, we would like to predict the rating that the user has not yet rated. So, we can recommend users the movies that they would like to give a high rating. In this recommendation system, we can get a matrix that the existing rating can be represented in. The matrix's size is $|U| \times |D|$. U means we have a set of users, and D means a set of movies. R_{ij} means the value of user i rated movies j. The basic idea of matrix factorization is finding out two matrics P(a $|U| \times K$ matrix) and Q(a $|D| \times K$ matrix) that their product approximates R:

$$R \approx P \times Q^T = \hat{R} \tag{1}$$

Where K is the number of latent features. In this way, each row of P would represent the strength of the associations between a user and the features. Similar to this, each row of Q would represent the strength of the associations between an item and the features. The prediction of a rating of an item d_j by u_i can calculate by:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{K}^{k=1} p_{ik} q_{kj} \tag{2}$$

The recommendation problem can be treated as a matrix completion problem. This model was proposed by Koren[5], and it has become the most common methods in recommendation system.

There are many different extensions to the matrix factorization algorithm, such as non-negative matrix factorization (NMF) that all the elements of the factor matrces (P and Q in the above example) should be non-negative[8]. But there still is a problem in MF based recommendation system. Although system prefer to recommending the item that has a high predictive score, people do not always consume those high-rating items.

B. Back-propagation nerual network

Rumelhart et al. [9] developed the back-propagation neural network (BPNN, see Fig.1).

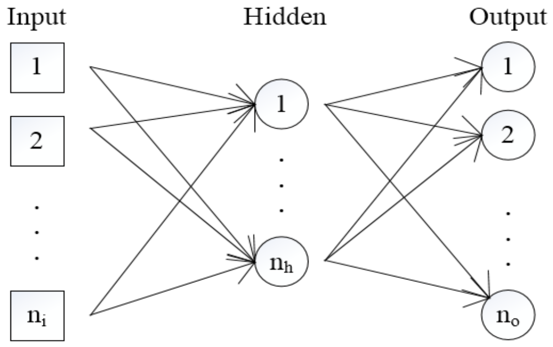


Fig. 1. Back-propagation neural network

It ia a solution to the problem of training multi-layer perceptions. The fundamental advances represented by the BPNN were the inclusion of a differentiable transfer function at each node of the network and the use of error back-propagation to modify the internal network weights after each training epoch. In this paper, BPNN is applied to predict each user would watch which video in short-term. Tsung-Lin Lee[10] applied BPNN to predict the short-term typhoon surge and surge deviation in order to overcome the problem of exclusive and nonlinear relationships. They proposed an alternative BPNN methodology in forecasting the short-term typhoon surge. The inputs to the BPNN are several typhoon parameters, namely, the wind speed, wind direction and the values of astronomical tide. And the output is tidal level.

C. RNN based recommendation

Netflix applied RNN to solve real-life recommend problems[11]. They proposed an RNN-based approach for session-based recommendation, and considered practical aspects of the task and introduces several modification to classic RNNs. Each user session in an e-commerce system can be modeled as a sequence of web page. Netease proposed a model that refreshes the recommendation result each time when user opens a new web page. In general, recurrent neural network uses traditional $tanh$ unit. Gated recurrent unit (GRU) was proposed by Cho et al. [12] to make each recurrent unit to adaptively capture dependencies of different time scales. And the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cells.

III. RNN for short-term recommendation

A. Short-term recommendation

In this paper, we define short-term recommendation as recommending user item that one will consume next time. We produce a prediction one item that user most likely to consume. For example, if user u_i has watched $DieHard$ 1, $DieHard$ 2 and $DieHard$ 3, we speculate that the next time he/she is most likely to watch $DieHard$ 4. We only focus on the short-term behavior of users. Through the analysis of the user's recent consuming behavior, we calculate the user the next most likely consuming item and recommend to him/her.

B. BPNN theory

A typical three-layered network with an input layer (I), a hidden layer (H) and an output layer (O) as shown in Figure 1. Each layer consists of several neurons and the layer are connected by sets of correlation weights. The neurons receive input from initial inputs. Suppose there is a training set (x^1,y^1),...,(x^m,y^m), we will write $a_i^{(l)}$ to denotes the activation of unit i in the layer l. For l=1 (input layer), $a_i^{(1)}$=x_i denote the i-th input. Given a fixed setting of the parameters W,b, the computation that neural network represents is given by:

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \tag{3}$$

$$a^{(l+1)} = f(z^{(l+1)}) \tag{4}$$

And output layer is softmax layer.

$$o^l = softmax(a^l) \tag{5}$$

Where f is an adequate nonlinear transfer function. A common transfer function is the sigmoid function expressed by $f_x = (1 + e^{-x})^{-1}$ which has a characteristic of $\mathrm{d}f/\mathrm{d}x = f(x)[1 - f(x)]$.

C. RNN theory

A recurrent neural network can be treated as a sequence, every unit's structure is the same, each unit can pass information to the next one. Fig. 2 is an unrolled recurrent network.

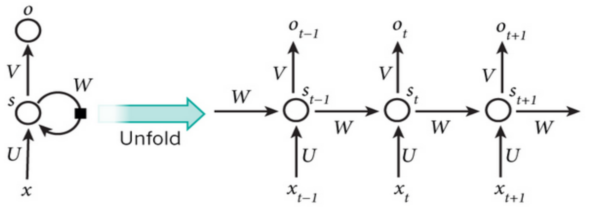


Fig. 2. An unrolled recurrent neural network

By unrolling we simply mean that we write out the network for the complete sequence. For example, if a user u_i watched five movies, the network would be unrolled into a 5-layer neural network, one layer for each movie. RNNs update the hidden state s_t using the following update function:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{6}$$

Where function f usually is a nonlinearity such as tanh or ReLu. x_t is the input at time step t, for example, x_1 could be a one-hot vector corresponding to the third movie of user's watch history. And we can compute o_t :

$$o_t = softmax(Vs_t) \tag{7}$$

It's obvious that RNNs will use all of the previous units' information to the present task. However, the effect of different positions on the state of current hidden layers is different, that is, each previous state is weighted by the distance. And when generating errors, the error may be caused by one or several units, it should be only corresponding units' weight to be updated. That's a improved version of RNNs, gated recurrent neural networks (GRUs). The state of hidden unit is a linear combination of previous activation and candidate activation $\hat{h}$

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h_t} \tag{8}$$

And we can update gate by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{9}$$

Where the candidate activation function $\hat{h}$ and the reset gate can be compute by:

$$\hat{h} = tanh(Wx_t + U(r_t \odot h_{t-1})) \tag{10}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{11}$$

D. Training MF model

In general, the rating matrix is high-dimensional and extremely sparse, and traditional singular value decomposition (SVD) methods only decompose dense matrix[13]. It means if we want to use SVD method, we should fill rating matrix at first. However, it will cause some problems, such as:

- It increases the amount of data and improves the complexity of the algorithm.
- It is easy to cause data distortion.

These problems lead to the traditional matrix decomposition method is not satisfactory. We use a matrix decomposition method that only takes into account existing rating, named Funk-SVD[14]. We only need to optimize the function:

$$min_{p^*,q^*} \sum_{u,i \in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{12}$$

Where P and Q have been defined before in related work, K is the set of (u,i) pairs of the existing rating, r_{ui} is the rating of the user u for item i, and $\|q_i\|^2 + \|p_u\|^2$ is the regularization to prevent overfitting, λ is the regularization coefficient.

We use stochastic gradient descent (SGD) to optimize previous objective function. Stochastic gradient descent, also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions. In other words, SGD tries to find minimums or maximums by iteration[5]. By training the MF model, we obtain the optimal parameters to make the score prediction more accurate.

E. Training BPNN

In this paper, we build a three-layer BPNN, it includes input layer, hidden layer and output layer.

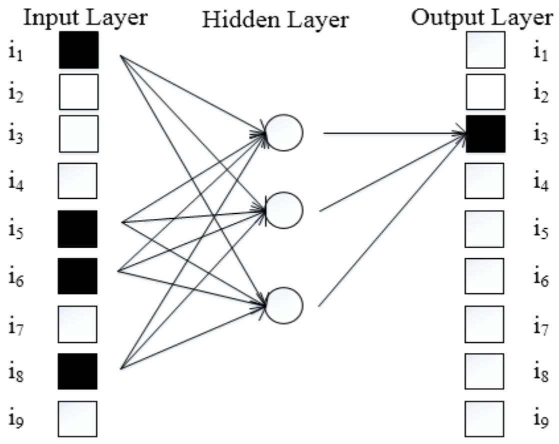


Fig. 3. BPNN training model

Suppose that we have a set of user (u_1, u_2), and they had watch a list of movies (u_1:(i_1, i_5, i_6, i_8,i_3), u_2:(i_2, i_4, i_6, i_7,i_1, i_3)) that sorted by time. We define n_len is the length of input sequence. Suppose n_l=4, the training data ($input$:$output$) will be ((i_1, i_5, i_6, i_8:i_3),(i_2, i_4, i_6, i_7:i_1),(i_4, i_6, i_7,i_1:i_3)). We will train each set of data sequentially. Fig. 3 shows the first set of data training, input data is (i_1, i_5, i_6, i_8), output data is (i_3).We will set the value of unit to one if it exists in input data.

The j-th unit's activation of hidden layer can be calculate by:

$$h_j = f(\sum_{i=1}^{m} W^1_{ij}x_i + b^1_j) \tag{13}$$

Where W^1 is the weight of the input layer to the hidden layer and b is the bias of the input layer. f is sigmoid or ReLu function.

For the output layer:

$$o_j = f(\sum_{i=1}^{k} W^2_{ij}h_i + b^2_j) \tag{14}$$

W^2 is the weight of the hidden layer to output layer and b^2 is the bias of the hidden layer.

The probability of selecting item i is as follows:

$$p_i = \frac{o_i}{\sum_{j=1}^{k} o_j} \tag{15}$$

If user consumes item i, we train the parameters in BPNN to maximize the probability of p_i.

F. Training RNN

It is known from above that:

$$s_t = tanh(Ux_t + Ws_{t-1}) \tag{16}$$

$$o_t = softmax(Vs_t) \tag{17}$$

We define our loss function to be the cross entropy loss, given by:

$$E_t(o_t, \hat{o}_t) = -o_t log\hat{o}_t \tag{18}$$

Where o_t is the correct movie at time step t, and $\hat{o}_t$ is our prediction. Before training, we set a number n_l first, n_l is the length of a training example. We use each user's historic data to generate training data. For example, for user u_i, if he watched 7 movies, and $n_l = 5$, then we can get the a set of training data:[$(x_0, x_1, x_2, x_3, x_4)$,$(x_1, x_2, x_3, x_4, x_5)$,$(x_2, x_3, x_4, x_5, x_6)$]. Fig. 4 shows the training model when $n_l = 5$.

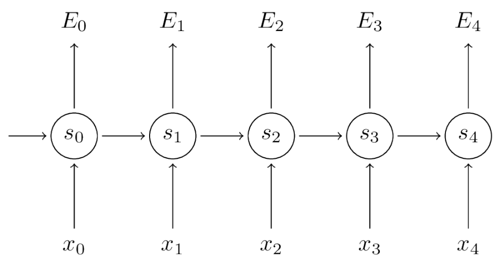


Fig. 4. A training model

In order to reduce the error, we calculate the gradients of the error with respect to parameters U,V and W and then learn good parameters using stochastic gradient descent.

We use back propagation algorithm to calculate these gradients. We will use E_2 as an example. The formula are as following:

$$\frac{\partial E_2}{\partial V} = \frac{\partial E_2}{\partial \hat{o_2}}\frac{\partial \hat{o_2}}{\partial V} = \frac{\partial E_2}{\partial \hat{o_2}}\frac{\partial \hat{o_2}}{\partial z_2}\frac{\partial z_2}{\partial V} = (\hat{o_2} - o_2) \otimes s_2 \tag{19}$$

In the above, $z_2 = Vs_2$, and $\otimes$ is the outer product of two vectors. For W:

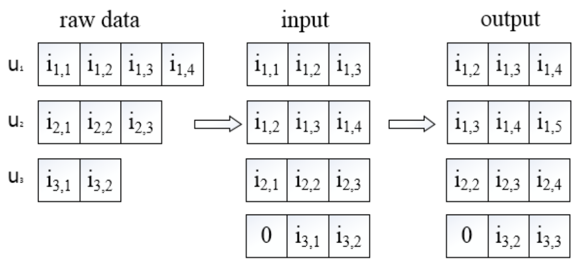
$$\frac{\partial E_2}{\partial W} = \frac{\partial E}{\partial \hat{o_2}}\frac{\partial \hat{o_2}}{\partial s_2}\frac{\partial s_2}{\partial W} \tag{20}$$

Where $s_2 = tanh(Ux_t + Ws_1)$ depends on s_1, which depends on W and s_0, and so on.

When training RNN, there are still some problems. Such as:

- The amount of movies n_i that the user u_i has watched is less than n_l.
- The vanishing gradient problem.

To solve the first problem, we take the approach shown in the Fig. 5.



raw data
input
output
u₁
u₂
u₃
$i_{1,1}$ $i_{1,2}$ $i_{1,3}$ $i_{1,4}$
$i_{2,1}$ $i_{2,2}$ $i_{2,3}$
$i_{3,1}$ $i_{3,2}$
$i_{1,1}$ $i_{1,2}$ $i_{1,3}$
$i_{1,2}$ $i_{1,3}$ $i_{1,4}$
$i_{2,1}$ $i_{2,2}$ $i_{2,3}$
0 $i_{3,1}$ $i_{3,2}$
$i_{1,2}$ $i_{1,3}$ $i_{1,4}$
$i_{1,3}$ $i_{1,4}$ $i_{1,5}$
$i_{2,2}$ $i_{2,3}$ $i_{2,4}$
0 $i_{3,2}$ $i_{3,3}$

Fig. 5. Input data processing

Fig. 6 shows that tanh and sigmoid functions have derivatives of 0 at both ends.

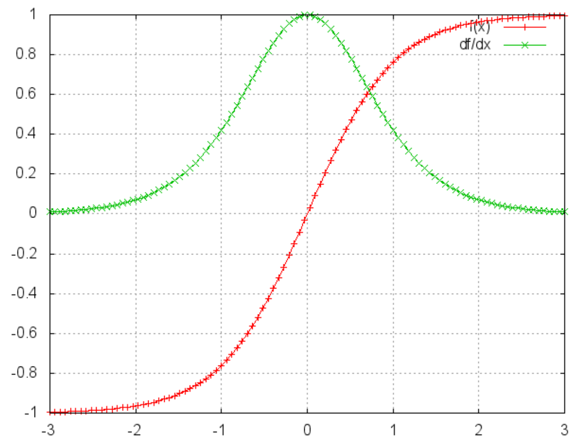


Fig. 6. tanh and derivative

It means that the corresponding neurons are saturated. The gradient is zero, and drives other gradients in previous layers towards 0. The gradient will vanish completely after a few steps when there are small values in matrix and multiple matrix multiplications. Thus, gradient contributions from far away steps become zero, and the state at those steps doesn't contribute to what you are learning.

G. Hybrid Recommendation Model

In traditional recommendation system, they often ignore the sequence of users' behavior. However, the sequence contains information about the users' interest migration. We used the GRU-based RNN in our models for short-term recommendation. The input of the network is the actual behavior of the users while the output is the item of the next event in short-term. In the input layer, the input vector's length equals to the number of items and only the coordinate corresponding to the active item is one, the others are zeros.

We find that the GRU shares the same output layer as the BPNN. The core of the network is that we will merge their output layers to produce unified result.

For the output (l-th) layer of BPNN, the i-th unit's activation $\bar{a}_i^{(l)}$ calculate by:

$$\bar{a}_i^{(l)} = f(\sum_{x=1}^{\bar{E}} W_i^{l-1} \bar{a}_x^{l-1} + b_x^{l-1}) \tag{21}$$

Where $\bar{E}$ is the number of unit in hidden layer.

For the output (l_0-th) layer of GRU, the i-th unit's activation $\bar{a_0}_i^{(l_0)}$ calculate by:

$$\bar{a_0}_i^{(l_0)} = f(\sum_{x=1}^{E} W_i^{l_0-1} \bar{a}_x^{l_0-1} + b_x^{l_0-1}) \tag{22}$$

Where E is the number of unit in one GRU's layer.

So, the predicted probability p_i of i-th item that user will consume can be calculate by:

$$p_i = \frac{f(\sum_{x=1}^{E} W_i^{l_0-1} \bar{a}_x^{l_0-1} + b_x^{l_0-1}) + f(\sum_{x=1}^{\bar{E}} W_i^{l-1} \bar{a}_x^{l-1} + b_x^{l-1})}{\sum_{j=1}^{N} f(\sum_{x=1}^{E} (W_j^{l_0-1} \bar{a}_x^{l_0-1} + b_x^{l_0-1}) + \sum_{x=1}^{\bar{E}} (W_j^{l-1} \bar{a}_x^{l-1} + b_x^{l-1}))} \tag{23}$$

In the above formula, $\bar{a_0}^{(l_0)}$ is the output value of GRU and $\bar{a_0}^{(l)}$ is the output value of BPNN. We did not manually set the coefficients to combine their results, but by the neural network training process to decide. The two networks can be trained independently.

As we all know, recommendation system not only need to consider the users' interest, we also recommend to users the popular items and the items that they might like. It is easy to find out the most popular items by the amount of items consumed. We also can calculate the rating for the items that the user didn't consume and recommend to users.

There is a trick, we pick out the last movie that each user watched from raw data. This ensures that all input data has a corresponding output data. Taking user's u_i raw data as an example, he has watched a set of movies (i_1, i_2, i_3, i_4), we will pick out i_4, so no matter how big n_l is, there always exists output data correspond to input data.

When using back propagation method to training RNN, each of the neural network's weights receives an update proportional to the gradient of the error function with respect to the current weight in each iteration of training. Traditional activation functions have gradients in the range (-1, 1) or [0, 1), and back propagation computes gradients by the chain rule. This has the effect of multiplying n of these small numbers to compute gradients of the front layers in an n-layer network, meaning that the gradient (error signal) decreases exponentially with n and the front layers train very slowly.

IV. Experiments and Discussion

We designed an experiment to predict what the next movie would look like, based on what the user had seen in the past. In short-term recommendation, we only need to recommend one movie to users.

A. Datasets and experiment design

We evaluated the above described model on MovieLens datasets, which are commonly used for evaluating recommendation system algorithm. The datasets (MovieLens 100K) consists 100,000 ratings (1-5) from 943 users on 1682 movies. Each user has rated at least 20 movies. And the data is randomly ordered. It's a tab separated list of user id, item id,rating and timestamp. The time stamps are unix seconds since 1/1/1970 UTC.

We sorted the data by timestamp. Then we divide the data into many parts. For MF model, we choose one part data to fill user-item matrix, and use MF method to calculate the rating. We could get the id (id_p) of movie that has highest score. We also could get the real id (id_r) of movie that the user will watch from datasets. If id_p equal to id_r, the prediction is correct. For RNN model, the input is the id of movie (one-hot encoding), and the output is softmax layer[15], [16], [17].

We classify the prediction problem as a multiclass classification problem. For multiclass classification with K >2, we apply a softmax nonlinearity in the output layer of K classes. The softmax function is defined as:

$$y_k = \frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}} \tag{24}$$

$\sum_{k'=1}^{K} y_k = 1$ and $y_k \in [0, 1]$ for all $1 \leq k \leq K$.

In the previous formula, a_k means the activation of k node.

B. Experiment comparison

We compare our model with a set of commonly methods.

- MP: Recommend to user with the most popular movie.
- KNN: We can calculate the similarity of all users. For user u_i, if the similarity between user u_i and user u_j is the highest, we will recommend to user u_i the movie that has the highest rating in the movies set that user u_j watched but u_i didn't.
- MF: The matrix factorization model as said in previous.
- RNN: RNN model.
- GRU: GRU model.
- BP-GRU: GRU model combine with BP model.

The most popular movies refer to movies that are the most viewed by all users.

For MF model, we used 10%,20%,...90% of the data for experiment. However, MF model is ineffective in short-term recommendation.

In our RNN/GRU model, there are four layers. The input data is a vector, the vector's length is the number of movies and only the coordinate corresponding to the movie that he/she watched is one, the others are zeros. And the Fig. 7 shows our model when the input data is (i_1,i_5,i_6,i_8), and the output data is (i_3). At first time step, user u_j watched the movie i_1, so the input data of first unit is $[1,0,0,0,0,0,0,0]$, and the output data of first unit is $[0,0,0,0,1,0,0,0]$.

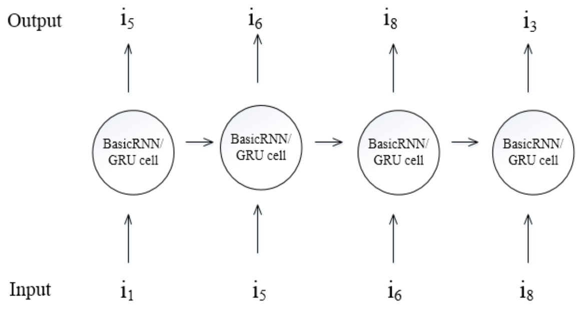


Fig. 7. RNN/GRU model

We compared the precision when the number of layers are 3-7, and Table 1 shows the result.

Table 1. Precision for different number of layers.

$Numberoflayers$	RNN	GRU
3	3.2%	5.3%
4	3.0%	6.3%
5	3.0%	4.7%
6	2.7%	3.2%
7	2.3%	3.0%

BP-GRU model is shown in Fig. 8, the length of input data n_l=4, there is one hidden layer in BPNN and four layers in RNN. In our experiment, the RNN's cell is GRU cell. As shown in Fig. 8, the input data is (i_1,i_5,i_6,i_8), and the output data is (i_3).

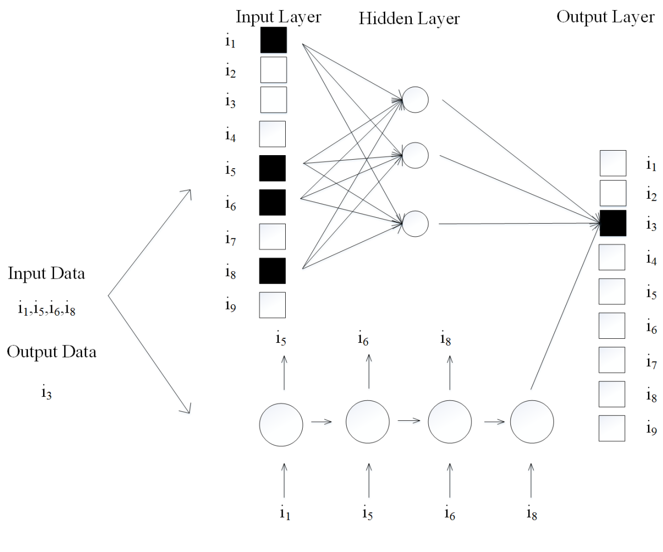


Fig. 8. BP-RNN model

We compared the GRU/BP/BP-GRU model's result under different number of layers as shown in Table 2.

Table 2. GRU/BP/BP-GRU model's precision for different number of layers.

$Numberoflayers$	GRU	BP	$BP-GRU$
3	5.3%	0.8%	5.8%
4	6.3%	1.2%	6.9%
5	4.7%	0.4%	4.9%
6	3.2%	0.2%	3.2%
7	3.0%	0%	3.0%

Table 3 shows that BP-GRU model get the highest precision. We also found that traditional recommend algorithms are ineffective in short-term recommendation. By using RNN, GRU and BP-GRU, precision has been significantly improved. The BP-GRU based approach has substantial gain over the RNN and GRU in precision.

Table 3. Precision for different models.

$Model$	$Precision$
MP	0%
KNN	0%
MF	0%
RNN	4.2%
GRU	6.3%
$BP-GRU$	%

In addition, we know that the number of hidden layer units also has an effect on the results. Fig. 9 shows the effect of sequence length on the precision.

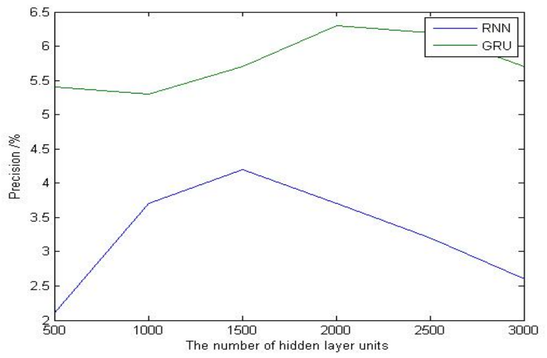


Fig. 9. RNN vs GRU

Fig. 9 shows the comparison of RNN and GRU. It's obvious that GRU is better than RNN. Although the accuracy is increased, the training time is increased accordingly.

V. Conclusion and Future Work

In this paper, we proposed a model for short-term recommendation, we use the users' history as an input to the RNN. After training, we can get the item that user would consume. In addition, to acquire better performance, we also tried some improvements such as combining RNN model with BP model. Experimental study has been conducted on commonly used datasets to prove that models are effective and get a higher accuracy compared with other methods.

However, there are still many issues worth considering, for example the rating matrix is sparse and coefficient expansion. To solve that, we can find out a new method to improve matrix factorization method. In addition, we can share some parameters to reduce the number of parameters.

Acknowledgment

The authors would like to thank the National Natural Science Foundation of China under Grant No.61503073, the Science Foundation of Heilongjiang Province of China for returned scholars under Grant No. LC2015025, the Y-outh Research Fund Project of Jilin Province under Grant No.20130522171JH, the Fundamental Research Funds for Central University under Grant No. HEUCF160602, and Harbin Special Fund for innovative talents of science and technology research under Grant No. 2013RFQXJ113.

References

[1] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Found. Trends Hum.-Comput. Interact.*, vol. 4, no. 2, pp. 81–173, Feb. 2011.

[2] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv e-prints*, vol. abs/1412.3555, 2014.

[3] M. Buscema, "Back propagation neural networks." vol. 33, pp. 233–270, 1998.

[4] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach," pp. 473–480, 2000.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[6] M. Tomas, K. Martin, B. Lukas, C. Jan, and K. Sanjeev, "Recurrent neural network based language model," *Interspeech*, pp. 1045–1048, 2010.

[7] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl, "Movielens unplugged: Experiences with an occasionally connected recommender system," pp. 263–266, 2003.

[8] N. Bertin, R. Badeau, and E. Vincent, "Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription," *Trans. Audio, Speech and Lang. Proc.*, vol. 18, no. 3, pp. 538–549, Mar. 2010.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," pp. 696–699, 1988.

[10] T.-L. Lee, "Back-propagation neural network for the prediction of the short-term storm surge in taichung harbor, taiwan," *Eng. Appl. Artif. Intell.*, vol. 21, no. 1, pp. 63–72, Feb. 2008.

[11] L. B. D. T. Balzs Hidasi, Alexandros Karatzoglou, "Session-based recommendations with recurrent neural networks," *CoRR*, vol. abs/1511.06939, 2015.

[12] D. B. Y. B. KyungHyun Cho, Bart van Merrienboer, "On the properties of neural machine translation: Encoder-decoder approaches." *CoRR*, vol. abs/1409.1259, 2014.

[13] J. V. Lieven De Lathauwer, Bart De Moor, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, pp. 1253–1278, 2000.

[14] M. Avriel, *Nonlinear Programming: Analysis and Methods*, ser. Dover Books on Computer Science Series. Dover Publications, 2003.

[15] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar 1994.

[16] Y. B. Razvan Pascanu, Tomas Mikolov, "On the difficulty of training recurrent neural networks." *International Conference on Machine Learning*, pp. 1310–1318, 2013.

[17] G. E. H. Ruslan Salakhutdinov, "Replicated softmax: an undirected topic model." *Annual Conference on Neural Information Processing Systems*, pp. 1607–1614, 2009.