Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A new item-based deep network structure using a restricted Boltzmann machine for collaborative filtering*

Yong-ping DU[†1], Chang-qing YAO[†‡2], Shu-hua HUO[1], Jing-xuan LIU[1]

([1]*Institute of Computer Science, Beijing University of Technology, Beijing 100124, China*)

([2]*Institute of Scientific and Technical Information of China, Beijing 100038, China*)

[†]E-mail: ypdu@bjut.edu.cn; yaocq@istic.ac.cn

Received Nov. 22, 2016; Revision accepted Mar. 7, 2017; Crosschecked Apr. 27, 2017

Abstract: The collaborative filtering (CF) technique has been widely used recently in recommendation systems. It needs historical data to give predictions. However, the data sparsity problem still exists. We propose a new item-based restricted Boltzmann machine (RBM) approach for CF and use the deep multilayer RBM network structure, which alleviates the data sparsity problem and has excellent ability to extract features. Each item is treated as a single RBM, and different items share the same weights and biases. The parameters are learned layer by layer in the deep network. The batch gradient descent algorithm with minibatch is used to increase the convergence speed. The new feature vector discovered by the multilayer RBM network structure is very effective in predicting a rating and achieves a better result. Experimental results on the data set of MovieLens show that the item-based multilayer RBM approach achieves the best performance, with a mean absolute error of 0.6424 and a root-mean-square error of 0.7843.

Key words: Restricted Boltzmann machine; Deep network structure; Collaborative filtering; Recommendation system
http://dx.doi.org/10.1631/FITEE.1601732 **CLC number:** TP391

1 Introduction

With the explosive growth of information, the application of recommender systems has attracted increasing attention (Schafer *et al.*, 1999; Shi *et al.*, 2011). The collaborative filtering (CF) method has been widely used as an effective recommendation technique. By analyzing historical user behavior, such as rating data and purchasing behavior, CF recommends items in which users may be interested. The effectiveness of the algorithm relies heavily on the historical user behavior data.

There are two types of CF algorithms: neighbor-based and model-based algorithms. Neighbor-based CF (Sarwar *et al.*, 2001; Linden *et al.*, 2010) calculates the similarity between users or items to accomplish the recommendation task, whereas model-based CF learns the user behavior data to build a model and then uses it to complete the recommendation. In recent years, there have been two popular model based CF algorithms: matrix factorization based (Koren *et al.*, 2009) and restricted Boltzmann machine (RBM) based. Ference *et al.*(2013) introduced a personalized location recommendation method based on CF. Li *et al.* (2013) integrated the results of the latent Dirichlet allocation (LDA) algorithm into the singular value decomposition (SVD) based CF algorithm and proposed an academic paper recommendation algorithm.

Despite the development of the CF technique, some problems still exist, such as data sparsity and cold start. Some methods have been proposed to alleviate the effect on system performance. Candès and Recht (2009) proposed the convex optimization algorithm to predict and fill the missing data in the

‡ Corresponding author
* Project supported by the National Science and Technology Support Plan (No. 2013BAH21B02-01) and the Beijing Natural Science Foundation (No. 4153058)
ORCID: Yong-ping DU, http://orcid.org/0000-0001-6867-2063

rating matrix, which solves the data sparsity problem. The voting method has also been used to predict missing data (Breese *et al.*, 1998). Zhang *et al*. (2014) put forward a semisupervised cotraining algorithm to solve the cold start problem.

There are several public data sets used to evaluate the performance of recommendation systems. GroupLens (Resnick *et al*., 1994) first introduces an automated collaborative filtering system using a neighborhood-based algorithm. The developers also published the MovieLens data set, which has been widely used by many research institutions. Our approach is also implemented on this data set. Nathanson *et al.* (2007) developed the Jester recommendation system, which improves the principal component analysis algorithm to accomplish a dimension reduction of the rating matrix, and achieves better performance. Also, the hierarchical structure in recommender systems is important, and Wang *et al.* (2015) proposed a recommendation framework to incorporate explicit hierarchical structures for recommendation and verified the effectiveness.

RBM was first introduced by Smolensky (1986). With the appearance of the RBM fast-learning algorithm known as contrastive divergence, RBMs have attracted an increasing amount of attention in the field of machine learning. The research foundation for deep learning was first introduced by Hinton and Sejnowski (1983). They proposed the training approach in a Boltzmann machine; however, parameter learning within the neural network was very slow and it affected the further development of this research. Hinton *et al.* (2006) introduced a greedy, layer-by-layer unsupervised learning algorithm that consists of learning a stack of RBMs, one layer at a time. After greedy learning, the entire stack could be viewed as a single probabilistic model known as a deep belief network (DBN). This type of deep structure achieves excellent performance for classification and recognition, and it outperforms other models, such as support vector machine (SVM) and AdaBoost. The theory of deep learning has recently been widely used in many machine learning tasks, such as object and speech recognition (Bengio, 2009; Mohamed *et al*., 2012).

Salakhutdinov *et al.* (2007) introduced a technique to apply an RBM for CF for the first time and achieved good performance on the Netflix data set (http://www.netflix.com), which was published for the development of CF algorithms in 2006. Feuerverger *et al.* (2012) provided a discussion and overview of some lessons for statistics that emerged from the Netflix contest. Many key ideas proposed by a large number of individuals and teams were analyzed. BellKor won the Netflix Grand Prize 2007 (Bell *et al.*, 2007). The Progress Prize 2008 was won by the combined efforts of BellKor and BigChaos (Töscher and Jahrer, 2008). Luo (2011) proved that the theory of RBM-based CF is based on the prototype user, and a recommendation was implemented by calculating the similarity between each user and the prototype user. Wu (2010) described how to integrate time bias and user bias into RBM-based CF to improve performance. However, existing RBM-based CF algorithms are primarily user based; thus, they treat each user as a single RBM. We propose an item-based RBM for CF and use the deep structure of a multi-RBM for parameter learning. The final rating data is predicted by the more accurate features obtained from the deep structure of RBM. Experimental results show that the multilayer RBM structure achieves the best performance when compared with both two-layer user-based and item-based RBMs.

2 Deep structure model using RBM

2.1 Basic model of RBM

An RBM is an undirected graph model, as shown in Fig. 1, where $\boldsymbol{v}$ is the visible layer for observed data and $\boldsymbol{h}$ is the hidden layer for feature extraction. There is a full connection between the hidden units and visible units. Each unit has a bias. The weight for each edge is labeled using vector $\boldsymbol{W}$. All the display units v_i and hidden units h_j are binary variables with the value of 0 or 1.

Suppose the RBM contains n visible units and m hidden units. Vectors $\boldsymbol{v}$ and $\boldsymbol{h}$ represent the states of the visible units and hidden units, respectively. Here, v_i denotes the state of visible unit i and h_j represents the state of hidden unit j. The energy of the RBM in Fig. 1 is defined as follows:

$$E(\boldsymbol{v},\boldsymbol{h}\,|\,\boldsymbol{\theta})=-\sum_{i=1}^{n}a_i v_i-\sum_{j=1}^{m}b_j h_j-\sum_{i=1}^{n}\sum_{j=1}^{m}v_i W_{ij}h_j, \quad (1)$$

where $\boldsymbol{\theta}=(W_{ij},\ a_i,\ b_j)$ are parameters of the RBM model, W_{ij} denotes the weight between visible unit i and hidden unit j, a_i represents the bias of visible unit i,

and b_j represents the bias of hidden unit j. The joint probability distribution is represented as

$$P(\boldsymbol{v},\boldsymbol{h}\,|\,\boldsymbol{\theta})=\frac{\exp\big(-E(\boldsymbol{v},\boldsymbol{h}\,|\,\boldsymbol{\theta})\big)}{Z(\boldsymbol{\theta})}, \tag{2}$$

where

$$Z(\boldsymbol{\theta})=\sum_{\boldsymbol{v},\boldsymbol{h}}\exp\big(-E(\boldsymbol{v},\boldsymbol{h}\,|\,\boldsymbol{\theta})\big) \tag{3}$$

is the normalization factor, which is also known as the partition function.

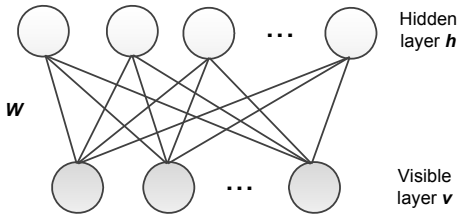

Fig. 1 Graphic model of a restricted Boltzmann machine (RBM)

Our main concern is the likelihood function $P(\boldsymbol{v}|\boldsymbol{\theta})$, which is the probability distribution of visible layer $\boldsymbol{v}$. It can be calculated as follows using the marginal probability distribution $P(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta})$:

$$P(\boldsymbol{v}\,|\,\boldsymbol{\theta})=\frac{1}{Z(\boldsymbol{\theta})}\sum_{\boldsymbol{h}}\exp\big(-E(\boldsymbol{v},\boldsymbol{h}\,|\,\boldsymbol{\theta})\big). \tag{4}$$

Here, $Z(\boldsymbol{\theta})$ should be calculated first and its time complexity is 2^{n+m}.

Because of the special structure of RBM, where the connection exists only between different layers, the state of the hidden layer units (or visible layer units) is independent when the state of the visible layer units (or hidden layer units) is given.

The equations

$$P(h_j=1\,|\,\boldsymbol{v})=\sigma\left(b_j+\sum_{i=1}^{n}v_iW_{ij}\right) \tag{5}$$

and

$$P(v_i=1\,|\,\boldsymbol{h})=\sigma\left(a_i+\sum_{j=1}^{m}h_jW_{ij}\right) \tag{6}$$

set a probability of 1 for the hidden layer unit and visible layer unit, respectively, where $\sigma(\cdot)$ is the sigmoid function.

2.2 Contrastive divergence learning

The main task of RBM model learning is to obtain the parameters that can be achieved by the likelihood function shown in Eq. (4). We can learn the optimal parameters using the method of stochastic gradient ascent, which yields the maximum value of $L(\boldsymbol{\theta})=\sum_{t=1}^{T}\log P(\boldsymbol{v}^{(t)}\,|\,\boldsymbol{\theta})$. The important step involves determining how to calculate each parameter's partial derivative. The following equation determines the result of the partial derivative (Zhang *et al*., 2015):

$$\begin{aligned}\frac{\partial L}{\partial\boldsymbol{\theta}}=\sum_{t=1}^{T}\Bigg(&\left\langle\frac{\partial(-E(\boldsymbol{v}^{(t)},\boldsymbol{h}\,|\,\boldsymbol{\theta}))}{\partial\boldsymbol{\theta}}\right\rangle_{P(\boldsymbol{h}|\boldsymbol{v}^{(t)},\boldsymbol{\theta})}\\&-\left\langle\frac{\partial(-E(\boldsymbol{v},\boldsymbol{h}\,|\,\boldsymbol{\theta}))}{\partial\boldsymbol{\theta}}\right\rangle_{P(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})}\Bigg),\end{aligned} \tag{7}$$

where T represents the total number of data and $<\cdot>_P$ represents the expected value under distribution P.

The first item in Eq. (7) can be calculated directly, and the second item can only be approximated using a sample method, such as the Gibbs sample, because of the existence of the normalization factor. Hinton (2002) provided a fast way to learn the RBM, namely, contrastive divergence (CD). The CD algorithm initializes the first step of the Gibbs sample as a data point and sets the state of the visible units to a sample point. Then, the state of the hidden units is calculated using Eq. (5). Additionally, the state of the visible units can be obtained using Eq. (6), and it produces the reconstruction of the visible units. A better approximate distribution is achieved after k steps of reconstruction. The term $P(\boldsymbol{h}|\boldsymbol{v}^{(t)}, \boldsymbol{\theta})$ represents the data, and 'recon' represents the distribution of the reconstruction. Each parameter of the RBM is updated as follows (Zhang *et al*., 2015):

$$\begin{cases}\Delta W_{ij}=\lambda(<v_ih_j>_{\text{data}}-<v_ih_j>_{\text{recon}}),\\ \Delta a_i=\lambda(<v_i>_{\text{data}}-<v_i>_{\text{recon}}),\\ \Delta b_j=\lambda(<h_j>_{\text{data}}-<h_j>_{\text{recon}}).\end{cases} \tag{8}$$

2.3 Deep structure built by RBM

Based on the two-layer RBM, we add multiple hidden layers in the network, and the connections exist only between adjacent layers that compose the

deep RBM structure shown in Fig. 2. The deep model can learn the internal representations, which capture the complicated structure in the higher layers.

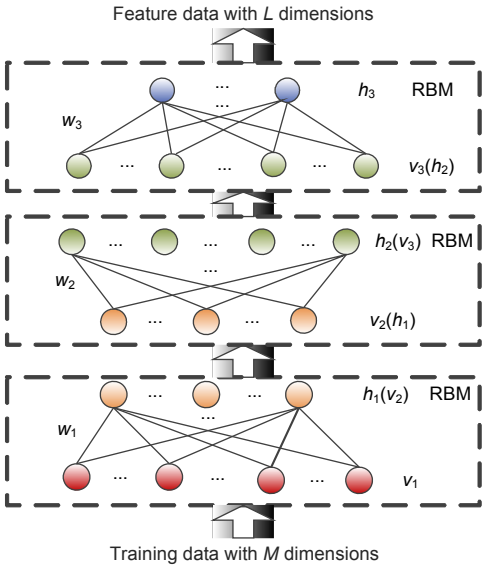

Fig. 2 Pretraining phase of the deep Boltzmann machine

Here, $\boldsymbol{\theta}=(w_1, w_2, w_3)$ are the model parameters to be learned in the three-hidden-layer structure. The clamped training data is used in the first layer of the RBM, and the weight value w_1 is learned after several iterations. At the same time, the hidden layer data h_1 is obtained and used as the visible layer data v_2 of the next higher RBM. To summarize, the hidden layer h_i of the lower RBM in the deep network structure is used as the visible layer v_{i+1} of the higher RBM. By using the same learning algorithm provided in Section 2.2, the weights w_2 and w_3 are also learned in the second and third RBMs, respectively.

The vector dimension is reduced from M to L using the deep structure shown in Fig. 2. We show that the new feature vector discovered by the deep Boltzmann machine is very effective in providing a rating prediction for CF.

3 Deep network structure for CF

3.1 Data representation in CF

Recommender systems are designed to predict user preferences for items that can be equally transformed into a matrix-filling problem. The rating data and example used in the system are shown in Fig. 3.

As shown in Fig. 3, user 1 and user 3 do not have rating data for the first and third items, respectively. The recommender system can provide a prediction for the missing values using the rating prediction model to fill the matrix. The prediction data for the missing values for user 1 and user 3 are four and three, respectively.

3.2 User-based RBM for CF

The neurons of the RBM network are random neurons and their output has just two states, namely, inactive and active (Smolensky, 1986). Two problems exist when applying RBM to CF. First, the units of RBM take on only values 1 and 0, and it should be solved to use the two-value variable to model the rating data, whose value is an integer. Second, users generally tend to rate a very small number of items; thus, the rating matrix is sparse.

To solve these problems, Salakhutdinov *et al.* (2007) provided a technique that uses an RBM for CF (Fig. 4). Suppose the rating data value is an integer between 1 and K, and there are n users and m items in the data set. We construct the visible layer with m softmax units and each softmax unit consists of k two-value subunits. Each user corresponds to a single RBM, and different users share the same weights and

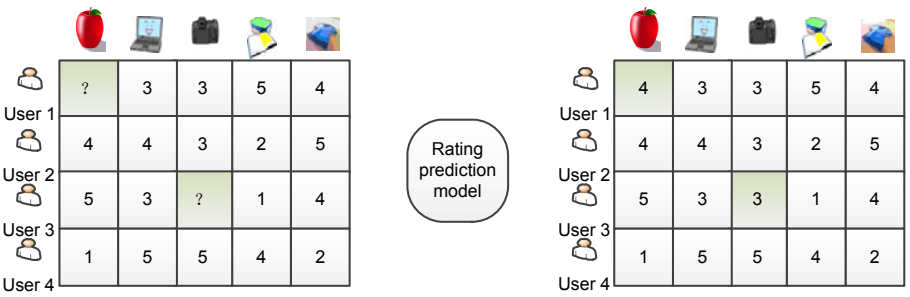

Fig. 3 Example of the user-item rating matrix and prediction matrix

biases. If the rating data of one user for an item is r ($1\leq r\leq K$), the rth subunit in the corresponding softmax is set to 1, and the other subunits' values are set to 0. This approach considers each user as a single RBM, and every softmax unit in the visible layer represents an item. We call this method a user-based RBM for CF.

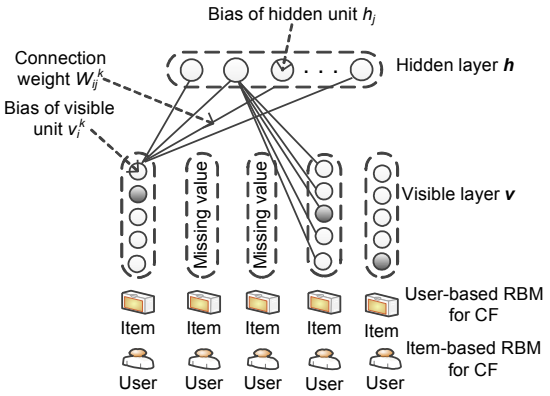

Fig. 4 Restricted Boltzmann machine (RBM) for collaborative filtering (CF), in which the visible layer can represent the item that denotes the user-based RBM for CF and, on the other hand, it represents the user, which denotes the item-based RBM for CF

3.3 Item-based RBM for CF

Most RBM-based CF methods are based on the user. We introduce an item-based RBM approach as shown in Fig. 4. For the visible layer in the RBM, every softmax unit represents a user, and each item corresponds to a single RBM. Different items share the same weights and biases. If a user does not provide rating data for an item, the RBM model for this item will not contain the softmax unit of this user.

The energy of the RBM model is calculated as

$$E(\boldsymbol{v},\boldsymbol{h}\,|\,\boldsymbol{\theta})=-\sum_{i=1}^{m}\sum_{j=1}^{F}\sum_{k=1}^{K}W_{ij}^{k}h_j v_i^k+\sum_{i=1}^{m}\log Z_i -\sum_{i=1}^{m}\sum_{k=1}^{K}v_i^k b_i^k-\sum_{j=1}^{F}h_j b_j, \tag{9}$$

where $Z_i=\sum_{k=1}^{K}\exp\left(b_i^k+\sum_j h_j W_{ij}^k\right)$ is the normalization factor, m is the user number or softmax unit number, F is the number of hidden units, and K denotes the range of rating data. Unrated user–item combinations are excluded from this expression.

The equations

$$P(v_i^k=1\,|\,\boldsymbol{h})=\frac{\exp\left(b_i^k+\sum_{j=1}^{F}h_j W_{ij}^k\right)}{\sum_{l=1}^{K}\exp\left(b_i^l+\sum_{j=1}^{F}h_j W_{ij}^l\right)} \tag{10}$$

and

$$P(h_j=1\,|\,\boldsymbol{v})=\sigma\left(b_j+\sum_{i=1}^{m}\sum_{k=1}^{K}v_i^k W_{ij}^k\right) \tag{11}$$

represent the conditional probability of setting visible unit's state and hidden unit's state to 1, respectively.

3.3.1 Learning parameters

The value of each parameter is updated as

$$\begin{cases}\Delta W_{ij}^k=\lambda_{\mathrm{w}}(<v_i h_j>_{\mathrm{data}}-<v_i h_j>_{\mathrm{recon}}),\\ \Delta b_i^k=\lambda_{\mathrm{v}}(<v_i>_{\mathrm{data}}-<v_i>_{\mathrm{recon}}),\\ \Delta b_j=\lambda_{\mathrm{h}}(<h_j>_{\mathrm{data}}-<h_j>_{\mathrm{recon}}),\end{cases} \tag{12}$$

where λ_{w}, λ_{v}, and λ_{h} are the updated weights for the three parameters. There is a small difference relative to the basic RBM. The users who do not provide a rating for an item are ignored when the visible layer units are reconstructed by the Gibbs sample.

We can apply the stochastic gradient descent (SGD) method to update the parameters. We choose SGD with minibatch to increase the convergence speed. For details, refer to Algorithm 1.

Algorithm 1 Training algorithm of item-based RBM for collaborative filtering

Input: training data set D_{train}, hidden unit number F, and iteration number n.
Output: value of parameters.

```
for (loop=0; loop<n; loop++)
  for all minibatch items in D_train
    for all items i in the current minibatch
      clamp all users who have rated this item on visible units
      compute p_j=P(h_j=1|v, θ) for all hidden units
      reconstruct the softmax units for the current item
      update p_j=P(h_j=1|v, θ) for all hidden units based on the reconstruction
    end for
    update all of the parameters by Eq. (12)
  end for
end for
```

For the multilayer RBM, the parameter learning process can be made more efficient by using a layer-by-layer pretraining phase as shown in Fig. 2. The parameters of $\boldsymbol{\theta}$=(w_1, w_2, w_3) in the three-hidden-layer structure are learned gradually (Fig. 5).

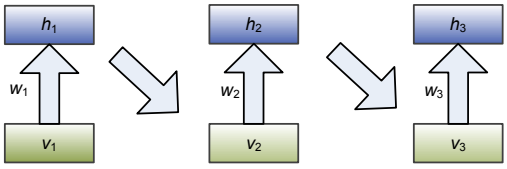

Fig. 5 Layer-by-layer parameter learning

The hidden layer data h_1 is used as the visible layer data v_2 in the second RBM, and h_2 is used as v_3 in the third RBM.

3.3.2 Rating prediction

After obtaining the parameter value during the training process, we can predict unknown ratings for the two-layer RBM using the mean field method. The details of the method are shown in Algorithm 2.

Algorithm 2 Rating prediction with item-based RBM for collaborative filtering

Input: user u, item i.

Output: rating data by user u for item i.

1. Clamp the users who have rated item i over the softmax units of the RBM model.

2. Compute $\hat{p}_j = p(h_j = 1 \mid \boldsymbol{v})$ for all hidden units j.

3. Compute

$$p(v_i^k = 1 \mid \hat{p}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} \hat{p}_j W_{ij}^k\right)}{\sum_{l=1}^{K} \exp\left(b_i^l + \sum_{j=1}^{F} \hat{p}_j W_{ij}^l\right)} \quad \forall\, k=1, 2, \ldots, K.$$

4. Obtain the prediction $\bar{r}_{ui} = \sum_{k=1}^{K} p(v_i^k = 1 \mid \hat{p})k$.

For the multilayer RBM, we obtain a new vector with L dimensions to represent each item after dimension reduction using the deep structure shown in Fig. 2, which is more effective.

4 Experimental results

4.1 Data sets and evaluation metrics

The experiments were implemented on MovieLens (http://www.grouplens.org), which provides rating data for movies. We used two data sets of size 100k and 1M, which are summarized in Table 1. The data sets were divided into two parts, and 80% was used as the training set.

Table 1 MovieLens data set in the experiment

Data set	User number	Movie number	Rating number
MovieLens-100k	1682	943	100 000
MovieLens-1M	6040	3952	1 000 209

We used the metrics mean absolute error (MAE)

$$\mathrm{MAE} = \frac{1}{|\mathrm{Test}|} \sum_{u,i \in \mathrm{Test}} \left| r_{ui} - r'_{ui} \right| \tag{13}$$

and root-mean-square error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{u,i \in \mathrm{Test}} (r_{ui} - r'_{ui})^2}{|\mathrm{Test}|}} \tag{14}$$

to evaluate the performance of the recommendation algorithm. Here, r_{ui} denotes the real rating and r'_{ui} the predicted rating. The smaller the MAE and RMSE values, the better the performance of the algorithm.

4.2 Performance comparison of two-layer user-based RBM and item-based RBM

We conducted an experiment to verify the effectiveness of the item-based RBM method on two data sets: MovieLens-100k and MovieLens-1M. The number of hidden units was set to 10, 20, …, 200. The performance comparison between the methods of user-based RBM and item-based RBM is shown in Figs. 6 and 7. MAE and RMSE were used as the evaluation metrics.

As shown in Figs. 6 and 7, item-based RBM achieved better performance than user-based RBM with a different number of hidden units on the MovieLens-100k data set. We also found that both methods achieved their best performance when the number of hidden units was set to 160 and 180 for the MovieLens-100k and MovieLens-1M data sets, respectively. Ultimately, item-based RBM achieved the best RMSE and MAE values of 0.8688 and 0.6841 on MovieLens-1M, respectively.

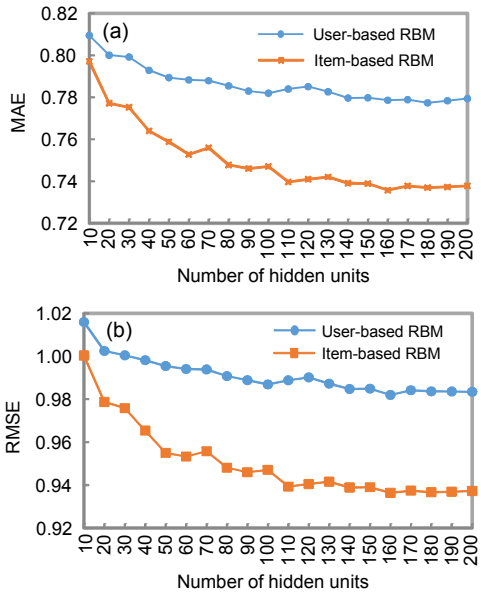

Fig. 6 Performance comparison of user-based RBM and item-based RBM on MovieLens-100k: (a) MAE; (b) RMSE

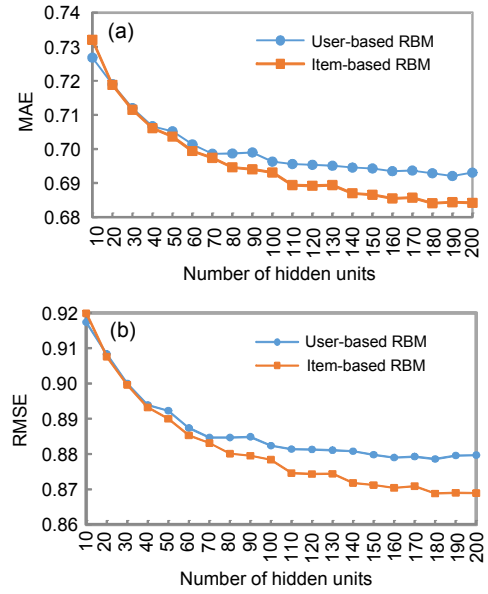

Fig. 7 Performance comparison of user-based RBM and item-based RBM on MovieLens-1M: (a) MAE; (b) RMSE

To verify the significance of the improvement in system performance, we provide the results of the one-sided *t*-test in Table 2. The item-based RBM and user-based RBM methods were compared using the evaluation metrics MAE and RMSE. We found that the *p*-values were both smaller than 0.05, which demonstrates that our approach for item-based RBM outperformed that for user-based RBM significantly.

Table 2 The results of *t*-test for the improvement of system performance (d*f*=28)

Metric*	*t*	*p*
MAE	8.52	1.45E–9
RMSE	9.41	2.56E–10

* User-based RBM vs. item-based RBM

Salakhutdinov *et al.* (2007) applied RBM to the Netflix data set and achieved an RMSE value of ~0.91. Luo (2011) gave a new explanation to the RBM model by applying user-based collaborative filtering. It introduced local user similarity and global user similarity methods and obtained an MAE value of 0.72 on the MovieLens data set.

4.3 Performance of the deep structure with multi-layer item-based RBM

The system achieved better performance when a deep learning method was used for CF. We implemented rating prediction using item-based multilayer RBM. The experimental results on two data sets are shown in Table 3, where *L* denotes the feature vector dimension after reduction by the multilayer RBM shown in Fig. 2; *L* can be regarded as the feature number. We found that the system exhibited better performance when *L* was set to 70 and 110 on the MovieLens-100k and MovieLens-1M data sets, respectively. In particular, the best results MAE=0.6424 and RMSE=0.7843 were yielded on MovieLens-1M.

Table 3 Performances of deep structure on two MovieLens data sets

L	MAE		RMSE	
	Movie-Lens-100k	Movie-Lens-1M	Movie-Lens-100k	Movie-Lens-1M
30	0.6874	0.6734	0.8546	0.8159
50	0.6733	0.6556	0.8457	0.8064
70	**0.6721**	0.6558	**0.8416**	0.7892
90	0.6725	0.6457	0.8418	0.7849
110	0.6821	**0.6424**	0.8454	**0.7843**
130	0.6779	0.6453	0.8452	0.7845

L: feature vector dimension

We also compared the performances of two-layer RBM and multilayer RBM for CF. The experimental results on the two data sets are shown in Table 4. The deep structure achieved better performance than both user-based and item-based two-layer RBMs.

Table 4 Comparison of performances of the two-layer and multilayer RBMs

RBM	MAE		RMSE	
	Movie-Lens-100k	Movie-Lens-1M	Movie-Lens-100k	Movie-Lens-1M
Two-layer (user-based)	0.7774	0.6921	0.9820	0.8786
Two-layer (item-based)	0.7371	0.6841	0.9373	0.8688
Multilayer (item-based)	**0.6721**	**0.6424**	**0.8416**	**0.7843**

4.4 Discussion

Data sparsity is the main problem for recommendation systems, especially for high-dimensional data. There are also some algorithms that use the matrix decomposition approach to obtain the low-dimensional feature vector; however, it is difficult to capture the effective feature. We adopted a deep structure based RBM model to determine the important latent features. The model achieved better performance when compared with both the two-layer user-based and item-based RBMs.

After the mapping by the deep structure RBM, high-dimensional data in the lower layer was mapped to the low-dimensional space in the higher layer, which provided features that are more abstract. System performance was improved after data mapping, which demonstrates that the deep network structure has excellent ability to select features and represent data. Additionally, the time consumed by the deep structure was greater because of its complex model structure, which requires further research on parallel computing.

From the experimental results, we find that item-based RBM for CF outperforms user-based RBM for CF. The important reason is that the item-based approach considers the historical interest of the user and it recommends items similar to what the user liked before, which is more comprehensible. The recommendation is more relevant to the user's current behavior. Usually, the recommendation by a small number of credible neighbors is more confident than that by many indistinguishable neighbors. The size of the item is smaller than the size of the user in the MovieLens data sets, and thus item-based RBM is more appropriate. In addition, it resolves the problem of the new user. The recommendation result can be generated only after one click by the new user.

5 Conclusions

We proposed a new item-based approach to apply RBM for collaborative filtering, and the multilayer deep structure was trained for rating prediction. It considers each item as a single RBM and different items share the same weights and biases. The stochastic gradient descent method with minibatch was used to update the parameters. All the parameters were learned layer by layer in the deep RBM structure. The experimental results on the MovieLens data set show that item-based RBM outperforms user-based RBM significantly. In particular, the multilayer RBM structure achieved the best performance relative to both the two-layer user-based and item-based RBMs.

The approach will be optimized further in our future work. The current algorithm includes only one bias in the RBM model. We will consider other biases to obtain better recommendations.

References

Bell, R., Koren, Y., Volinsky, C., 2007. The BellKor Solution to the Netfix Prize. http://netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

Bengio, Y., 2009. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, **2**(1):1-127. http://dx.doi.org/10.1561/2200000006

Breese, J., Hecherman, D., Kadie, C., 1998. Empirical analysis of predictive algorithms for collaborative filtering. Proc. 14th Conf. on Uncertainty in Artificial Intelligence, p.43-52.

Candès, E.J., Recht, B., 2009. Exact matrix completion via convex optimization. *Found. Comput. Math.*, **9**(6):717-772. http://dx.doi.org/10.1007/s10208-009-9045-5

Ference, G., Ye, M., Lee, W.C., 2013. Location recommendation for out-of-town users in location-based social networks. Proc. 22nd ACM Int. Conf. on Information & Knowledge Management, p.721-726. http://dx.doi.org/10.1145/2505515.2505637

Feuerverger, A., He, Y., Khatri, S., 2012. Statistical significance of the Netflix challenge. *Stat. Sci.*, **27**(2):202-231. http://dx.doi.org/10.1214/11-STS368

Hinton, G.E., 2002. Training products of experts by minimizing contrastive divergence. *Neur. Comput.*, **14**(8):1771-1800. http://dx.doi.org/10.1162/089976602760128018

Hinton, G.E., Sejnowski, J., 1983. Optimal perceptual inference. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, p.448-453.

Hinton, G.E., Osindero, S., Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. *Neur. Comput.*, **18**(7):1527-1554. http://dx.doi.org/10.1162/neco.2006.18.7.1527

Koren, Y., Bell, R., Volinsky, C., 2009. Matrix factorization techniques for recommender systems. *Computer*, **42**(8): 30-37. http://dx.doi.org/10.1109/MC.2009.263

Li, Y., Yang, M., Zhang, Z.M., 2013. Scientific articles recommendation. Proc. 22nd ACM Int. Conf. on Information & Knowledge Management, p.1147-1156. http://dx.doi.org/10.1145/2505515.2505705

Linden, G., Smith, B., York, J., 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Int. Comput.*, **7**(1):76-80. http://dx.doi.org/10.1109/MIC.2003.1167344

Luo, H., 2011. Restricted Boltzmann Machines: a Collaborative Filtering Perspective. PhD Thesis, Shanghai Jiao Tong University, Shanghai, China (in Chinese).

Mohamed, A., Dahl, G.E., Hinton, G.E., 2012. Acoustic modeling using deep belief networks. *IEEE Trans. Audio Speech Lang. Process.*, **20**(1):14-22. http://dx.doi.org/10.1109/TASL.2011.2109382

Nathanson, T., Bitton, E., Goldberg, K., 2007. Eigentaste 5.0: constant-time adaptability in a recommender system using item clustering. Proc. ACM Conf. on Recommender Systems, p.149-152. http://dx.doi.org/10.1145/1297231.1297258

Resnick, P., Iacovou, N., Suchak, M., *et al*., 1994. GroupLens: an open architecture for collaborative filtering of netnews. Proc. ACM Conf. on Computer Supported Cooperative Work, p.175-186. http://dx.doi.org/10.1145/192844.192905

Salakhutdinov, R., Mnih, A., Hinton, G.E., 2007. Restricted Boltzmann machines for collaborative filtering. Proc. 24th Annual Int. Conf. on Machine Learning, p.791-798. http://dx.doi.org/10.1145/1273496.1273596

Sarwar, B., Karypis, G., Konstan, J., *et al*., 2001. Item-based collaborative filtering recommendation algorithms. Proc. ACM 10th Int. Conf. on World Wide Web, p.285-295. http://dx.doi.org/10.1145/371920.372071

Schafer, J.B., Konstan, J., Riedl, J., 1999. Recommender systems in E-commerce. Proc. 1st ACM Conf. on Electronic Commerce, p.158-166. http://dx.doi.org/10.1145/336992.337035

Shi, J., Chen, J., Bao, Z., 2011. An application study on collaborative filtering in e-commerce. Int. Conf. on Service Systems and Service Management, p.1-7. http://dx.doi.org/10.1109/ICSSSM.2011.5959336

Smolensky, P., 1986. Information processing in dynamical systems: foundations of harmony theory. *In*: Rumellhart, D.E., McClelland, J.L. (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations. MIT Press, Cambridge, MA, USA, p.194-281.

Töscher, A., Jahrer, M., 2008. The BigChaos Solution to the Netfix Prize 2008. http://netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

Wang, S., Tang, J., Wang, Y., 2015. Exploring implicit hierarchical structures for recommender systems. Proc. 24th Int. Joint Conf. on Artificial Intelligence, p.1813-1819.

Wu, J.L., 2010. Collaborative Filtering on the Netflix Prize Dataset. PHD Thesis, Peking University, Beijing, China (in Chinese).

Zhang, C.X., Ji, N.N., Wang, G.W., 2015. Restricted Boltzman machines. *Chin. J. Eng. Math.*, **32**(2):159-173 (in Chinese). http://dx.doi.org/10.3969/j.issn.1005-3085.2015.02.001

Zhang, M., Tang, J., Zhang, X., *et al*., 2014. Addressing cold start in recommender systems: a semi-supervised co-training algorithm. Proc. 37th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.73-82. http://dx.doi.org/10.1145/2600428.2609599