Deep Matrix Factorization Models for Recommender Systems*

Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, Jiajun Chen
National Key Laboratory for Novel Software Technology; Nanjing University, Nanjing 210023, China
Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023, China
xuehj@nlp.nju.edu.cn, {daixinyu,zjb,huangsj,chenjj}@nju.edu.cn

Abstract

Recommender systems usually make personalized recommendation with user-item interaction ratings, implicit feedback and auxiliary information. Matrix factorization is the basic idea to predict a personalized ranking over a set of items for an individual user with the similarities among users and items. In this paper, we propose a novel matrix factorization model with neural network architecture. Firstly, we construct a user-item matrix with explicit ratings and non-preference implicit feedback. With this matrix as the input, we present a deep structure learning architecture to learn a common low dimensional space for the representations of users and items. Secondly, we design a new loss function based on binary cross entropy, in which we consider both explicit ratings and implicit feedback for a better optimization. The experimental results show the effectiveness of both our proposed model and the loss function. On several benchmark datasets, our model outperformed other state-of-the-art methods. We also conduct extensive experiments to evaluate the performance within different experimental settings.

1 Introduction

In the era of information explosion, information overload is one of the dilemmas we are confronted with. Recommender systems (RSs) are instrumental to address this problem as they help determine which information to offer to individual consumers and allow online users to quickly find the personalized information that fits their needs [Sarwar *et al.*, 2001; Linden *et al.*, 2003]. RSs are nowadays ubiquitous in e-commerce platforms, such as recommendation of books at Amazon, music at Last.com, movie at Netflix and reference at CiteULike.

Collaborative filtering (CF) recommender approaches are extensively investigated in research community and widely used in industry. They are based on the simple intuition that if users rate items similarly in the past, they are likely to rate other items similarly in the future [Sarwar *et al.*, 2001; Linden *et al.*, 2003]. As the most popular approach among various collaborative filtering techniques, *matrix factorization* (MF) which learns a latent space to represent a user or an item becomes a standard model for recommendation due to its scalability, simplicity, and flexibility [Billsus and Pazzani, 1998; Koren *et al.*, 2009]. In the latent space, the recommender system predicts a personalized ranking over a set of items for each individual user with the similarities among the users and items.

Ratings in the user-item interaction matrix are explicit knowledge which have been deeply exploited in early recommendation methods. Because of the variation in rating values associated with users on items, biased matrix factorization [Koren *et al.*, 2009] are used to enhance the rating prediction. To overcome the sparseness of the ratings, additional extra data are integrated into MF, such as social matrix factorization with social relations [Ma *et al.*, 2008; Tang *et al.*, 2013], topic matrix factorization with item contents or reviews text [McAuley and Leskovec, 2013; Bao *et al.*, 2014], and so on.

However, modeling only observed ratings is insufficient to make good top-N recommendations [Hu *et al.*, 2008]. Implicit feedback, such as purchase history and unobserved ratings, is applied in recommender systems [Oard *et al.*, 1998]. The SVD++ [Koren, 2008] model firstly factorizes the rating matrix with the implicit feedback, and is followed by many techniques for recommender systems [Rendle *et al.*, 2009; Mnih and Teh, 2012; He and McAuley, 2015].

Recently, due to the powerful representation learning abilities, deep learning methods have been successfully applied including various areas of Computer Vision, Audio Recognition and Natural Language Processing. A few efforts have also been made to apply deep learning models in recommender systems. Restricted Boltzmann Machines [Salakhutdinov *et al.*, 2007] was firstly proposed to model users' explicit ratings on items. Autoencoders and the denoising autoencoders have also been applied for recommendation [Li *et al.*, 2015; Sedhain *et al.*, 2015; Strub and Mary, 2015]. The key idea of these methods is to reconstruct the users' ratings through learning hidden structures with the explicit historical ratings. Implicit feedback is also applied in this research line of deep learning for recommendation. An extended work pre-

*Xin-Yu Dai is the corresponding author. This work was supported by the 863 program(2015AA015406) and the NSFC (61472183,61672277).

sented a collaborative denoising autoencoder (CDAE) [Wu *et al.*, 2016] to model user's preference with implicit feedback. Another work of neural collaborative filtering (NCF) [He *et al.*, 2017] was proposed to model the user-item interactions with a multi-layer feedforward neural network. Two recent works above exploit only implicit feedback for item recommendations instead of explicit rating feedback.

In this paper, to make use of both explicit ratings and implicit feedback, we propose a new neural matrix factorization model for top-N recommendation. We firstly construct a user-item matrix with both explicit ratings and non-preference implicit feedback, which is different from other related methods using either only explicit ratings or only implicit ratings. With this full matrix (explicit ratings and zero of implicit feedback) as input, a neural network architecture is proposed to learn a common latent low dimensional space to represent the users and items. This architecture is inspired by the deep structured semantic models which have been proved to be useful for web search [Huang *et al.*, 2013], where it can map the query and document in a latent space through multiple layers of non-linear projections. In addition, we design a new loss function based on cross entropy, which includes the considerations of both explicit ratings and implicit feedback.

In sum, our main contributions are outlined as follows.

- We propose novel deep matrix factorization models with a neural network that map the users and items into a common low-dimensional space with non-linear projections. We use a matrix including both explicit ratings and non-preference implicit feedback as the input of our models.
- We design a new loss function to consider both explicit ratings and implicit feedback for better optimization.
- The experimental results show the effectiveness of our proposed models which outperform other state-of-the-art methods in top-N recommendation.

The organization of this paper is as follows. Problem statement is introduced in Section 2. In Section 3, we present the architecture and details of the proposed models. In Section 4, we give empirical results on several benchmark datasets. Concluding remarks with a discussion of some future work are in the final section.

2 Problem Statement

Suppose there are M users $\mathcal{U} = \{u_1, ..., u_M\}$, N items $\mathcal{V} = \{v_1, ..., v_N\}$. Let $R \in \mathbb{R}^{M \times N}$ denote the rating matrix, where R_{ij} is the rating of user i on item j, and we mark unk if it is unknown. There are two ways to construct the user-item interaction matrix $Y \in \mathbb{R}^{M \times N}$ from R with implicit feedback as,

$$Y_{ij} = \begin{cases} 0, & if \quad R_{ij} = unk \\ 1, & otherwise \end{cases} \tag{1}$$

$$Y_{ij} = \begin{cases} 0, & if \quad R_{ij} = unk \\ R_{ij}, & otherwise \end{cases} \tag{2}$$

Most of the existing solutions for recommendation apply Equation 1 to construct the interaction matrix of Y [Wu *et al.*, 2016; He *et al.*, 2017]. They consider all observed ratings the same as 1. In this paper, we construct the matrix of Y with the Equation 2. The rating R_{ij} of user u_i on item v_j is still reserved in Y. We think that the explicit ratings in Equation 2 is non-trivial for recommendation because they indicate the preference degree of a user on an item. Meanwhile, we mark a zero if the rating is unknown, which is named as non-preference implicit feedback in this paper.

The recommender systems are commonly formulated as the problem of estimating the rating of each unobserved entry in Y, which are used for ranking the items. Model-based approaches [Koren, 2008; Salakhutdinov and Mnih, 2007] assume that there is an underlying model which can generate all ratings as follows.

$$\hat{Y}_{ij} = F(u_i, v_j|\Theta) \tag{3}$$

where $\hat{Y}_{ij}$ denotes the predicted score of interaction Y_{ij} between user u_i and item v_j, Θ denotes the model parameters, and F denotes the function that maps the model parameters to the predicted scores. Based on this function, we can achieve our goal of recommending a set of items for an individual user to maximize the user's satisfaction.

Now, the next question is how to define such a function F. Latent Factor Model (LFM) simply applied the dot product of p_i, q_j to predict the $\hat{Y}_{ij}$ as follows [Koren *et al.*, 2009]. Here, p_i and q_j denote the latent representations of u_i and v_j, respectively.

$$\hat{Y}_{ij} = F^{LFM}(u_i, v_j|\Theta) = p_i^T q_j \tag{4}$$

Recently, neural collaborative filtering (NCF) [He *et al.*, 2017] presented an approach with a multi-layer perceptron to automatically learn the function of F. The motivation of this method is to learn the non-linear interactions between users and items.

In this paper, we follow the Latent Factor Model which uses the inner product to calculate the interactions between users and items. We do not follow the neural collaborative filtering because we try to get the non-linear connection between users and items through a deep representation learning architecture.

We give the notations used in the following section. u indicates a user and v indicates an item. i and j index u and v, respectively. Y denote the user-item interaction matrix transformed by Equation 2, Y^+ denotes the observed interactions, Y^- means all zero elements in Y and $Y^-_{sampled}$ denotes the set of negative instances, which can be all (or sampled from) Y^-. Then $Y^+ \cup Y^-_{sampled}$ means all training interactions. We denote the i-th row of matrix Y by Y_{i*}, j-th column by Y_{*j} and its $(i, j) - th$ element by Y_{ij} .

3 Our Proposed Model

In this section, we firstly briefly introduce the deep structure semantic model which inspires us to propose our method. Then, we present our proposed architecture to represent the users and items in a latent low-dimensional space. Lastly, we give our designed loss function for optimization, followed by the model training algorithm.

3.1 Deep Structure Semantic Model

Deep Structured Semantic Models (DSSM) were proposed in [Huang *et al.*, 2013] for web search. It uses a deep neural network to rank a set of documents for a given query. DSSM firstly maps the query and the documents to a common lower semantic space with a non-linear multi-layer projection. And then for web search ranking, the relevance of query to each document is calculated by cosine similarity between the low dimensional vectors of query and document. The deep neural network are discriminatively trained to maximize the conditional likelihood of the query and matched documents.

DSSM has been applied for users modeling [Elkahky *et al.*, 2015]. Different from our work, it focused on modeling the user with rich extra features, such as the web browsing history and search queries. We only use the observed ratings and observed feedback since we focus on the traditional top-N recommendation problem.

3.2 Deep Matrix Factorization Models (DMF)

As mentioned in Section 2, we form a matrix Y according to the Equation 2. With this matrix Y as the input, we propose an architecture of deep neural network to project users and items into a latent structured space. Figure 1 illustrates our proposed architecture.

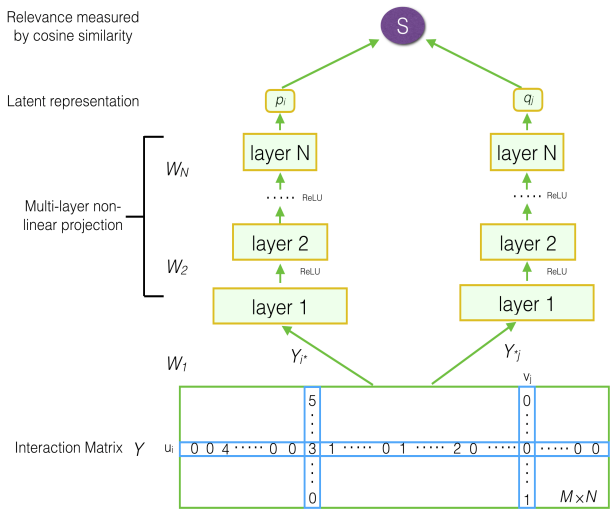

Figure 1: The architecture of Deep Matrix Factorization Models

From the matrix Y, each user u_i is represented as a high-dimensional vector of Y_{i*}, which represents the i_{th} user's ratings across all items. Each item v_j is represented as a high-dimensional vector of Y_{*j}, which represent the j_{th} item's ratings across all users. In each layer, each input vector is mapped into another vector in a new space. Formally, if we denote the input vector by x , the output vector by y , the intermediate hidden layers by $l_i, i = 1, ..., N-1$, the i_{th} weight matrix by W_i, the i_{th} bias term by b_i, and the final output latent representation by h. We have

$$\begin{aligned} l_1 &= W_1 x \\ l_i &= f(W_{i-1} l_{i-1} + b_i), i = 2, ..., N-1 \\ h &= f(W_N l_{N-1} + b_N) \end{aligned} \tag{5}$$

where we use the $ReLU$ as the activation function at the output layer and the hidden layers $l_i, i = 2, ..., N-1$:

$$f(x) = max(0, x) \tag{6}$$

In our architecture, we have two multi-layer networks to transform the representations of u and v respectively. Through the neural network, the user u_i and item v_j are finally mapped to a low-dimensional vector in a latent space as shown in Equation 7. The similarity between the user u_i and item v_j is then measured according to the Equation 8.

$$\begin{aligned} p_i &= f_{\theta_N^U}(...f_{\theta_3^U}(W_{U2} f_{\theta_2^U}(Y_{i*} W_{U1}))...) \\ q_j &= f_{\theta_N^I}(...f_{\theta_3^I}(W_{V2} f_{\theta_2^I}(Y_{*j}^T W_{V1}))...) \end{aligned} \tag{7}$$

Here W_{U1} and W_{V1} are the first layer weighting matrix for U and V, respectively, and W_{U2} and W_{V2} for the second layer, and so on.

$$\hat{Y}_{ij} = F^{DMF}(u_i, v_j | \Theta) = cosine(p_i, q_j) = \frac{p_i^T q_j}{\|p_i\| \, \|q_j\|} \tag{8}$$

In our architecture, besides the multi-layers representation learning, we want to emphasize again that, to our best knowledge, it is the first time to use the interaction matrix directly as the input for representation learning. As we mentioned before, Y_{i*} represents a user's ratings across all items. It can to some extent indicate a user's global preference. And Y_{*j} represents an item's ratings by all users. It can to some extent indicate an item's profile. We believe that these representations of users and items are very useful for their final low-dimensional representations.

3.3 Loss Function

Another key component for recommendation models is to define a proper objective function for model optimization according to the observed data and unobserved feedback.

A general objective function is as follows.

$$L = \sum_{y \in Y^+ \cup Y^-} l(y, \hat{y}) + \lambda \Omega(\Theta) \tag{9}$$

Where $l(\cdot)$ denotes a loss function and $\Omega(\Theta)$ is the regularizer.

For recommender systems, two types of objective functions are commonly used, point-wise and pair-wise, respectively. For simply, we use point-wise objective function in this paper, and leave the pair-wise version to our future work.

Loss function is the most important part in the objective function. Squared loss is largely performed in many existing models [Salakhutdinov and Mnih, 2007; Koren *et al.*, 2009; Ning and Karypis, 2011; Hu *et al.*, 2008].

$$L_{sqr} = \sum_{(i,j) \in Y^+ \cup Y^-} w_{ij} (Y_{ij} - \hat{Y}_{ij})^2 \tag{10}$$

where w_{ij} denotes the weight of training instance (i, j). The use of the squared loss is based on the assumption that observations are generated from a Gaussian distribution [Salakhutdinov and Mnih, 2007]. However, the square loss can not be

used well with implicit feedback, because for implicit data, the target value Y_{ij} is a binarized 1 or 0 denoting whether i has interacted with j or not. In what follows, a loss function which pays special attention to the binary property of implicit data was proposed by [He *et al.*, 2017] as follows.

$$L = - \sum_{(i,j) \in Y^+ \cup Y^-} Y_{ij} log \hat{Y_{ij}} + (1 - Y_{ij}) log(1 - \hat{Y_{ij}}) \quad (11)$$

This loss is actually the binary cross-entropy loss (briefly as ce), addressing the recommendation with implicit feedback as a binary classification problem.

In sum, square loss pays attention to explicit ratings, while cross entropy loss pays attention to implicit ratings. In this paper, we design a new loss function to incorporate the explicit ratings into cross entropy loss, so that explicit and implicit information can be used together for optimization. We name our new loss as normalized cross entropy loss (briefly as nce), which is presented in Equation 12.

$$\begin{aligned} L = - \sum_{(i,j) \in Y^+ \cup Y^-} (& \frac{Y_{ij}}{max(R)} log \hat{Y}_{ij} \\ & + (1 - \frac{Y_{ij}}{max(R)}) log(1 - \hat{Y}_{ij})) \end{aligned} \quad (12)$$

We use the $max(R)$ (5 in a 5-star system) for normalization which is the max score in all ratings, so that different values of Y_{ij} have different influences to the loss.

Algorithm 1 DMF Training Algorithm With Normalized Cross Entropy

Input: $Iter$: # of training iterations,
neg_ratio: Negative sampling ratio,
R: original rating matrix,
Output: W_{Ui}(i=1..N-1): weight matrix for user,
W_{Vi}(i=1..N-1): weight matrix for item,

Initialisation :
  randomly initialize W_U and W_V;
  set $Y \leftarrow$ use Equation 2 with R;
  set $Y^+ \leftarrow$ all none zero interactions in Y;
  set $Y^- \leftarrow$ all zero interactions in Y;
  set $Y^-_{sampled} \leftarrow$ sample $neg\_ratio * \|Y^+\|$ interactions from Y^-;
  set $T \leftarrow Y^+ \cup Y^-_{sampled}$;
for it from 1 to $Iter$ **do**
  for each interaction of User i and Item j in T **do**
    set p_i, $q_j \leftarrow$ use Equation 7 with input of Y_{i*}, Y_{*j};
    set $\hat{Y}^o_{ij} \leftarrow$ use Equation 8,13 with input of p_i, q_j;
    set $L \leftarrow$ use Equation 11 with input of $\hat{Y}^o_{ij}$, Y_{ij};
    use back propagation to optimize model parameters
  end for
end for

3.4 Training Algorithm

For cross entropy loss, because the predicted score of Y_{ij} can be negative, we need to use Equation 13 to transform the original predictions. Let μ be a very small number, and we set $1.0e^{-6}$ in our experiments.

$$\hat{Y}^o_{ij} = max(\mu, \hat{Y}_{ij}) \quad (13)$$

We describe the detailed training method in Algorithm 1.

In Algorithm 1, we present the high-level training process of DMF model. For training the parameters of weight matrix W_U and W_V on each layer, we use back propagation to update the model parameters with batches. The complexity of our algorithm is linear to the size of matrix and the layers of network.

4 Experiments

In this section, we conduct experiments to demonstrate the effectiveness of both our proposed architecture and the refined loss function. We also do some extensive experiments to compare the performance with different experimental settings, such as the negative sampling ratio, number of layers in network, and so on.

4.1 Experimental Settings

Datasets

We evaluate our models on four widely used datasets in recommender systems: MovieLens 100K(ML100k), MovieLens 10M(ML1m), Amazon music(Amusic), Amazon movies(Amovie). They are publicly accessible on the websites [1] [2]. For MovieLens dataset we do not process it because it was already filtered, and for Amazon dataset we filtered the dataset, so that similar to the MovieLens data, only those users with at least 20 interactions and items with at least 5 interactions are retained [Wu *et al.*, 2016; He *et al.*, 2017]. The statistics of the four datasets are given in Table 1.

Statistics	ML100k	ML1m	Amusic	Amovie
# of Users	944	6,040	844	9,582
# of Items	1,683	3,706	18,813	92,221
# of Ratings	100,000	1,000,209	46,468	766,759
Rating Density	0.06294	0.04468	0.00292	0.00087

Table 1: Statistics of the Four Datasets

Evaluation for Recommendation

To evaluate the performance of item recommendation, we adopted the *leave-one-out* evaluation, which has been widely used in the literatures [He *et al.*, 2016; Kingma and Ba, 2014; He *et al.*, 2017]. We held-out the latest interaction as a test item for every user and utilize the remaining dataset for training. Since it is too time-consuming to rank all items for every user during evaluation, following [Koren, 2008; He *et al.*, 2017], we randomly sample 100 items that are not interacted by the users. Among the 100 items together with the test item, we get the ranking according to the prediction. We also use *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG) [He *et al.*, 2015] to evaluate the ranking performance. In our experiments, we truncated the ranked

[1] https://grouplens.org/datasets/movielens/
[2] http://jmcauley.ucsd.edu/data/amazon/

Datasets	Metrics	Methods				DMF		Improvement of
		ItemPop	ItemKNN	eALS	NeuMF-p	DMF-2-ce	DMF-2-nce	DMF-2-nce vs. NeuMF-p
ML100k	NDCG	0.231	0.334	0.356	0.395	0.405	0.409	3.5%
	HR	0.406	0.600	0.621	0.670	0.679	0.687	2.5%
ML1m	NDCG	0.263	0.372	0.425	0.440	0.442	0.451	2.5%
	HR	0.472	0.637	0.709	0.722	0.720	0.732	1.4%
Amusic	NDCG	0.242	0.345	0.374	0.371	0.403	0.397	7.0%
	HR	0.423	0.493	0.521	0.527	0.570	0.563	6.8%
Amovie	NDCG	0.386	0.403	0.455	0.512	0.533	0.550	7.4%
	HR	0.620	0.652	0.693	0.739	0.765	0.773	4.6%

Table 2: NDCG@10 and HR@10 Comparisons of Different Methods

list at 10 for both metrics. As such, the HR intuitively measures whether the test item is present on the top-10 list, and the NDCG measures the ranking quality which assigns higher scores to hits at top position ranks.

Detailed Implementation

We implemented our proposed methods based on Tensorflow[3], which will be released publicly upon acceptance. To determine hyper-parameters of DMF methods, we randomly sampled one interaction for each user as the validation data and tuned hyper-parameters on it. When training our models, we sampled seven negative instances per positive instance. For neural network, we randomly initialized model parameters with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01), optimizing the model with mini-batch Adam [Kingma and Ba, 2014]. We set the batch size to 256, and set the learning rate to 0.0001.

4.2 Performance Comparison

In this subsection, we compare the proposed DMF with the following methods. As our proposed methods aim to model the relationship between users and items, we mainly compare with user-item models. We leave out the comparison with item-item models, such as SLIM [Ning and Karypis, 2011] , CDAE [Wu *et al.*, 2016] because the performance difference may be caused by the user models for personalization. We also leave out the comparison with MV-DSSM [Elkahky *et al.*, 2015] because it uses a lot of auxiliary extra data and evaluates on its own datasets.

ItemPop It ranked the items by their popularity judged by the number of interactions. It is a non-personalized method whose performance is usually used as the baseline for personalized methods.

ItemKNN This is a standard item-based collaborative filtering method used by Amazon commercially [Sarwar *et al.*, 2001; Linden *et al.*, 2003].

eALS It is a state-of-the-art MF method for recommendation with square loss. It used all unobserved interactions as negative instances and weighted them non-uniformly by the item popularity. We tuned its hyper-parameters in the same way as [He *et al.*, 2016].

NeuMF-p This is a state-of-the-art MF method for item recommendation with cross entropy loss. It is the most related work to us. Different from our models, it only used the implicit feedback and initialized the representation of users and items randomly. After that, it leverages a multi-layer perceptron to learn the user-item interaction function. We name the neural matrix factorization with pretraining as NeuMF-p which showed the best performance among their proposal models. We tuned its hyper-parameters in the same way as [He *et al.*, 2017].

DMF-2-ce This is our proposed deep matrix factorization model, with 2 layers in the network and cross entropy as loss function. We use the matrix including the explicit ratings and implicit feedback as the input of DMF. We name this model as DMF-2-ce.

DMF-2-nce DMF-2-nce has the same depth of 2 layers in the network as that in DMF-2-ce except that it uses the normalized cross entropy loss.

The results of the comparison are summarized in Table 2. It demonstrate the effectiveness of both our proposed architecture and the loss function. As for the proposed architecture, on almost all datasets, both of our two models achieve the best performance in both metics of NDCG and HR, compared to other methods. Even compared to the state-of-the-art method of NeuMF-p, DMF-2-nce obtain 2.5-7.4% (5.1% average) and 1.4-6.8% (3.8% average) relative improvements in NDCG and HR metrics, respectively. As for the loss function, we compared the performances of our two models. DMF-2-nce achieves better results than DMF-2-ce, except on the dataset of Amusic.

4.3 Impact of the Input Matrix for DMF

		LFM-nce	DMF-1-nce
ML100k	NDCG	0.369	0.386
	HR	0.634	0.670
ML1m	NDCG	0.376	0.383
	HR	0.641	0.660
Amusic	NDCG	0.311	0.389
	HR	0.491	0.572
Amovie	NDCG	0.468	0.520
	HR	0.714	0.764

Table 3: Results for different input matrix. LFM-nce initialize the input matrix randomly. DMF-1-nce use the matrix of Y as input. They both perform 1-layer projection.

In DMF, we use the interaction matrix Y as the input. If we

[3] https://www.tensorflow.org

randomly initialize the representation vector of each user and each item as the input to a one layer DMF model, the model would be a standard Latent Factorization Model (LFM). To test the usefulness of the input matrix of Y, we conduct experiments on two models of LFM-nce and DMF-1-nce. They both have one layer in network and use the same loss function. From Table 3, we can observe that, with the input matrix, DMF-1-nce obtains a significant improvement over LFM-nce.

4.4 Sensitivity to Hyper-Parameters

		neg-1	neg-2	neg-5	neg-9	neg-10
ML100k	NDCG	0.393	0.403	0.406	0.401	0.400
	HR	0.677	0.687	0.684	0.675	0.667
ML1m	NDCG	0.408	0.432	0.434	0.443	0.438
	HR	0.689	0.721	0.723	0.726	0.725
Amusic	NDCG	0.384	0.387	0.386	0.391	0.384
	HR	0.569	0.562	0.556	0.567	0.554
Amovie	NDCG	0.521	0.541	0.549	0.548	0.544
	HR	0.767	0.778	0.781	0.774	0.776

Table 4: Results for Models with different negative sampling ratio.

Negative Sampling Ratio

In algorithm 1 as shown in Section 3.4, we need to sample negative instances from unobserved data for training. In this experiment, we apply different negative sampling ratio to observe the performance variance (e.g. neg-5 means we set the negative sampling ratio as 5). From the results in Table 4, we can find that more negative instances seem useful to improve performance. For these four datasets, the optimal negative sampling ratio is around 5 which is consistent with the results by previous work [He *et al.*, 2017].

Depth of Layers in Network

In our proposed model, we map the users and items to low-dimensional representations through neural network with multiple hidden layers. We conduct an extensive experiment on the Ml datasets to investigate our model with different number of hidden layers. For detailed comparison, Figure 2 shows the performance of each iteration by different layers. For space limitation, we just present the results on ML datasets. As shown in Figure 2, on the large ML1m dataset, our model with 2-layers illustrates the best performance. While on the relative small ML100k dataset, 2-layers almost gets the best performance, but not stably and significantly. Deeper layers seem not useful, and 3-layers model even decreases the performance.

Factors of the Final Latent Space

Besides the number of the hidden layers, the factors in each layer is possibly another sensitive parameter in our model. For simplicity, we just compare the performance with different number of factors on the top final latent space. We conduct the experiments to a two-layers model, and set the number of factors on the top layer from 8 to 128. As shown in Table 5, the final layer with 64 factors gets the best performance except on the dataset of Amusic. On the Amusic dataset, the best performance appears with 128 factors. The final representations with more factors might be more useful when the dataset is very sparse and small.

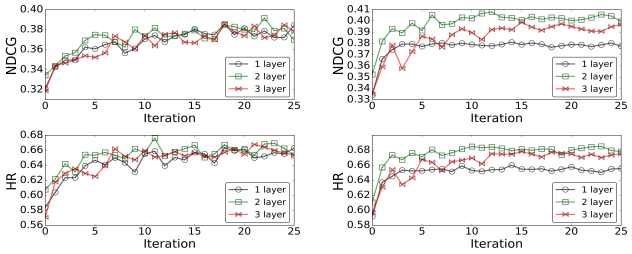

Figure 2: Results for models with different deep layers. Left: ML100k; Right: ML1m.

		8	16	32	64	128
ML100k	NDCG	0.369	0.389	0.386	0.394	0.393
	HR	0.660	0.672	0.675	0.682	0.677
ML1m	NDCG	0.361	0.398	0.406	0.411	0.408
	HR	0.637	0.681	0.688	0.690	0.689
Amusic	NDCG	0.357	0.371	0.377	0.374	0.384
	HR	0.547	0.560	0.568	0.559	0.569
Amovie	NDCG	0.485	0.514	0.522	0.524	0.521
	HR	0.740	0.763	0.767	0.768	0.767

Table 5: Results for models with different factors of the final latent space.

5 Conclusion and Future Work

In this paper, we propose a novel matrix factorization model with a neural network architecture. Through the neural network architecture, users and items are projected into low-dimensional vectors in a latent space. In our proposed model, we make full use of both explicit ratings and implicit feedback in two ways. The input matrix to our proposed model includes both explicit ratings and non-preference feedback. In another way, we also design a new loss function for training our models in which both explicit and implicit feedback are considered. The experiments on several benchmark datasets demonstrate the effectiveness of our proposed model.

In the future, there are two directions to extend our work. Pairwise objective function is another optional way for recommender system. We will verify our model with pairwise objective function. Because of the sparseness and large missing unobserved data, many works try to incorporate auxiliary extra data into recommender systems, such as social relation, review text, browsing history, and so on. This give us another interesting direction to extend our model with extra data.

References

[Bao *et al.*, 2014] Yang Bao, Hui Fang, and Jie Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, 2014.

[Billsus and Pazzani, 1998] Daniel Billsus and Michael J Pazzani. Learning collaborative information filters. In *ICML*, 1998.

[Elkahky *et al.*, 2015] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288. ACM, 2015.

[He and McAuley, 2015] Ruining He and Julian McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1510.01784*, 2015.

[He *et al.*, 2015] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1661–1670. ACM, 2015.

[He *et al.*, 2016] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558. ACM, 2016.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International World Wide Web Conference*, 2017.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

[Huang *et al.*, 2013] Po-Sen Huang, Xiaodong He, Jianfeng Gao, et al. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *In ICLR*, pages 1–15, 2014.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer, IEEE*, 42(8):30–37, 2009.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[Li *et al.*, 2015] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 811–820. ACM, 2015.

[Linden *et al.*, 2003] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 2003.

[Ma *et al.*, 2008] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: Social recommendation using probabilistic matrix factorization. In *CIKM*, 2008.

[McAuley and Leskovec, 2013] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.

[Mnih and Teh, 2012] Andriy Mnih and Yee W Teh. Learning label trees for probabilistic modelling of implicit feedback. In *Advances in Neural Information Processing Systems*, pages 2816–2824, 2012.

[Ning and Karypis, 2011] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 497–506. IEEE, 2011.

[Oard *et al.*, 1998] Douglas W Oard, Jinmook Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, pages 81–83, 1998.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, et al. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[Salakhutdinov and Mnih, 2007] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Nips*, volume 1, pages 2–1, 2007.

[Salakhutdinov *et al.*, 2007] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, et al. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[Sedhain *et al.*, 2015] Suvash Sedhain, Menon, et al. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM, 2015.

[Strub and Mary, 2015] Florian Strub and Jeremie Mary. Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *NIPS Workshop on Machine Learning for eCommerce*, 2015.

[Tang *et al.*, 2013] Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. Exploiting local and global social context for recommendation. In *IJCAI*, 2013.

[Wu *et al.*, 2016] Yao Wu, Christopher DuBois, et al. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 153–162. ACM, 2016.