Deep Matrix Factorization for Recommender Systems with Missing Data not at Random

Fei Zhang\textsuperscript{1a}, Jiaxing Song\textsuperscript{1b} and Shiyu Peng\textsuperscript{1c}

\textsuperscript{1}Department of Computer Science and Technology, Tsinghua University, Beijing, China
\textsuperscript{2}zhangfei15@mails.tsinghua.edu.cn, \textsuperscript{b}jxsong@tsinghua.edu.cn, \textsuperscript{c}psy16@mails.tsinghua.edu.cn

Abstract Matrix Factorization is a widely used collaborative filtering method in recommender systems. However, most of them are under the assumption that the rating data is missing at random (MAR), which may not be very common. For some users, they may only rate those movies they like, so the inferences will be biased in previous models. In this paper, we proposed a deep matrix factorization method based on missing not at random (MNAR) assumption. As far as we know, this model firstly uses deep learning method to address MNAR issue. The model consists of a complete data model (CDM) and a missing data model (MDM), which are both learned by neural networks. The CDM is nonlinearly determined by two factors, the user latent features and item latent features like other matrix factorization methods. And the MDM also use these two factors but taking the rating value as extra information while training. We used variational Bayesian inference to generate the posterior distribution of our proposed model. Through extensive experiments on different kind of datasets, our proposed model produce gains in some widely used metrics, comparing with several state-of-the-art models. We also explore the performance of our model within different experimental settings.

1. Introduction
In the era of information explosion, information overload makes it difficult for users to obtain their interested items and for service providers to process accurate item delivery. To overcome this problem, recommender systems are widely used, since they can help service providers to offer pertinent items to individual consumers. Nowadays, recommender systems are fundamental in various commerce sites, such as recommendation of movies at Netflix, commodities at Amazon and photos at Instagram.

Collaborative filtering (CF) recommender systems are the most representative and effective approaches in industry. Among various CF techniques, matrix factorization (MF) is the most popular one since Netflix Prize [1] [2]. MF aims at decomposing a user-item matrix into a low-dimensional space, and predicting unknown value by using the inner product of latent vectors. MF has been enhanced by many researchers in early works [3] [4]. And some methods are focus on modeling implicit feedback [3] [5] [6], since users can’t explicitly rate items in many scenarios. Recently, deep learning methods have been shown to improve performances of recommender systems [7]-[9]. MF methods are also adapted to take advantage of learning abilities of neural networks. The simplest way is concatenating latent vectors and their inner product as the input layer [10]. And neural collaborative filtering (NCF) use a multi-layer feedforward network to model user-item interactions [11].

Most of these MF methods are under the assumption that user-item interaction matrix is MAR [12]. That is, the process of selecting observed data is independent of the value of unobserved data.
data is MAR, the model trained by observed data can be applied to the whole data set. However, this assumption may be incorrect while considering following scenarios in a movie rating application: i) user may only watch movies that they like, and may only rate movies that they have watched; ii) user may rate movies with extreme bias. In these scenarios, the dependence between the observation process and the missing data can’t be ignored, and consequently the data is not MAR. An online survey organized by Yahoo Music also shows that the distribution of ratings for random songs was quite different with the distribution of voluntary ratings [13]. When the MAR assumption fails, the prediction accuracy of those inferences will be suffered. To address the lack of robustness of MAR, several adaptive traditional CF methods based on MNAR were proposed. The most common way to deal with MNAR data is to learn jointly a complete data model (CDM) and a missing data model (MDM) [12]. Some early works based on MNAR used traditional CF methods to train CDM and MDM [13]-[15]. However, these models are lacking of flexibility and they are out of date.

In this paper, we originally use deep learning techniques to train the CDM and MDM based on MNAR assumption. Firstly, we construct two deep matrix factorization models to obtain CDM and MDM with user-item rating matrix as input. By combining the MDM with the CDM, we build a joint model for prediction. A new loss function was also proposed by considering the accuracy of CDM and MDM at the same time. In summary, the contributions of our work are outlined as follows:

1) We employ novel deep matrix factorization to model MNAR data, which is premier to address MNAR issue by deep learning.
2) We design a new neural network to combine the CDM model and MDM model with elaborate inference. And a new loss function is applied while training the network.
3) We perform extensive experiments on open datasets, which indicates our model produce gains in various evaluation metrics.

2. Related Work
In this section, we first review the theory of missing data and discuss the common mechanisms under different assumption. Then, we will describe how to process deep matrix factorization by using Multilayer Perceptron (MLP).

2.1. Principles in Missing Data Model
The main principles to handle missing data are developed by Litter & Rubin [12]. We assume that our input data is the rating matrix \( R \in \mathbb{R}^{M \times N} \). Each element \( r_{ui} \), where \( u = 1, \ldots, M \) and \( i = 1, \ldots, N \), represents the rating value of user \( u \) on item \( i \). We can divide the matrix into two parts, \( R = \{ R^*, R^- \} \), where \( R^* \) and \( R^- \) denote the set of observed and missing entries in \( R \), respectively. And we also introduce a companion matrix \( X \), where \( x_{ui} \) in \( X \) is a Bernoulli random variable. The value of \( x_{ui} \) indicates whether \( r_{ui} \) is observed \( (x_{ui} = 1) \) or not \( (x_{ui} = 0) \). To obtain the matrix \( R \) and \( X \), we assume that \( R \) is generated by the CDM with parameters \( \Psi \) and \( X \) is generated by the MDM with parameters \( \Theta \). And both models may share a set of latent variables \( Z \). As such, the joint distribution for \( R, X \) and \( Z \) determined by \( \Psi \) and \( \Theta \) is

\[
p(X, R, Z | \Psi, \Theta) = p(X | R, Z, \Theta) p(R, Z | \Psi)
\]

There are three types of methods to process missing data. The most restrictive assumption is completely missing at random (CMAR), which refers the probability of observing a rating is independent of any ratings and latent variables, that is, \( p(X | R, Z, \Theta) = p(X | \Theta) \). The second assumption is MAR, which refers that the observation probability depends only upon the observed data and \( \Theta \), that is, \( p(X | R, Z, \Theta) = p(X | R^*, \Theta) \). In this sense, the MDM can be ignored when learning \( \Theta \), since \( p(X | R^*, \Theta) \) is constant with respect to \( \Theta \) [13]. However, the inferences will be biased if the MAR assumption is used when data is MNAR. To correct the observational bias, we must jointly learn CDM and MDM and predict unknown values by combining them together.
2.2. **Deep Matrix Factorization**

Matrix Factorization is a popular CF method which associates each user and item with a real-valued latent vector representing features. Formally, we let $\mathbf{p}_u$ denotes the latent vector of user $u$, and $\mathbf{q}_i$ denotes the latent vector of item $i$. Then we collect all the $\mathbf{p}_u$ to matrix $\mathbf{P} \in \mathbb{R}^{M \times K}$, which denotes the user latent matrix. Similarly, $\mathbf{Q} \in \mathbb{R}^{N \times K}$ is the item latent matrix. $K$ is the dimension of the latent space, and usually $K$ is far less than $M$ and $N$. In the traditional MF methods, the dot production of $\mathbf{p}_u$ and $\mathbf{q}_i$ is simply considered as the prediction of $\hat{r}_{ui}$, as shown by the following equation,

$$f(u, i | \mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^{K} p_{uk}q_{ik}$$

(2)

The inner production linearly combines each dimension of latent vector with same weight, which may limit the expression of MF [11]. To resolve the limitation, we can use MLP to capture the non-linear relationship between latent vectors.

![Figure 1. Framework of Deep Matrix Factorization](image)

Figure 1 illustrates the general framework of a MF methods using MLP. The bottom layer takes the features of user and item as input. These features can be generated by various models, such as content-based [16], neighbor-based and social-based. Since MF method is a CF method on rating matrix, it only takes the identities of user and item as input. Following the input layer, the embedding layer converts the sparse input features into dense latent vectors, same as the vector $\mathbf{p}_u$ and $\mathbf{q}_i$ mentioned above. After that, the latent vectors are fed into $H$ hidden layers, where each layer will capture the latent knowledge between user-item interactions. At last, the output layer will generate the predicted value $\hat{r}_{ui}$. To train the neural network, a loss function should be defined to the predicted value $\hat{r}_{ui}$ and target value $r_{ij}$, usually a pointwise loss function will be applied as following equation,

$$L = \sum_{u,i}(r_{ui} - \hat{r}_{ui})^2 + \lambda \left( \sum_u || \mathbf{p}_u ||^2 + \sum_i || \mathbf{q}_i ||^2 \right),$$

(3)

where $\lambda$ is a regularization hyperparameter. Several model extend this general framework, and train model on observed explicit or implicit feedback data. In our work, we adopt this general framework to train the CDM and MDM respectively.

3. **DMF-MNAR**

In this section, a probabilistic model based on DMF will be firstly introduced as our CDM. Then we describe MDM, which will generate the $\mathbf{X}$ based on $\mathbf{R}$. Finally, we will construct a joint model combining two models together.
3.1. The Complete Data Model
To build a complete data model with MF method, we assume that \( \mathbf{R} \) is generated by two low dimension matrix \( \mathbf{P} \) and \( \mathbf{Q} \). And each discrete rating \( r_{ui} \) is determined by the output of a MLP \( y_{ui} = f_{cdm}(\mathbf{p}_u, \mathbf{q}_i) \), where \( \mathbf{p}_u \) and \( \mathbf{q}_i \) are served as input. More precisely, the interaction in MLP can be defined as the following equations,

\[
\begin{align*}
    f_{cdm}(\mathbf{p}_u, \mathbf{q}_i) &= \phi_{out}(\phi_H(\phi_{H-1}(\cdots(\phi_1(\mathbf{p}_u, \mathbf{q}_i))))), \\
    \mathbf{t}_1 &= \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{W}_1 \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix} + \mathbf{b}_1, \\
    \phi_h(\mathbf{t}_{h-1}) &= f_{\sigma}(\mathbf{W}_h \mathbf{t}_{h-1} + \mathbf{b}_h), h = 2, 3, \ldots, H, \\
    \phi_{out} &= \sigma(\phi_H(\mathbf{t}_{H-1})),
\end{align*}
\]

where \( \mathbf{W}_h \), \( \mathbf{b}_h \) and \( \mathbf{t}_h \) denote the weight matrix, bias vector and output of the corresponding \( h \)-th layer of MLP, respectively. For the activation function \( f_{\sigma} \), it can be chosen among some basic functions, such as sigmoid, hyperbolic tangent (tanh) and Rectifier (ReLU). In this work, we opt ReLU as the activation function because it is more biologically plausible and proved to be non-saturated [17]; moreover, it is more suitable for spare data. To avoid the predictions outside of valid ratings, we can use logistic function \( \sigma(x) = 1/(1 + \exp(-x)) \) to bound the output into \( (0, 1) \). At same time, we map the ratings \( 1, 2, \ldots, V \) to the same interval by using the function \( g(x) = (x-1)/(V-1) \), where \( V \) is the maximum rating value.

When \( y_{ui} \) is given, to generate the \( r_{ui} \), we assume that the ratings are distributed under a Gaussian distribution with \( y_{ui} \) as mean value. And we also place zero-mean fully-factorized Gaussian priors \( \mathbf{P} \) and \( \mathbf{Q} \). So that, the distributions of \( \mathbf{R} \), \( \mathbf{P} \) and \( \mathbf{Q} \) are

\[
\begin{align*}
    p(\mathbf{R} | \mathbf{P}, \mathbf{Q}, \sigma^2_\mathbf{R}) &= \prod_{u=1}^{M} \prod_{i=1}^{N} N (r_{ui} | y_{ui}, \sigma^2_\mathbf{R}) \\
    p(\mathbf{P} | \sigma^2_\mathbf{P}) &= \prod_{u=1}^{M} N (\mathbf{p}_u | 0, \sigma^2_\mathbf{P}), \\
    p(\mathbf{Q} | \sigma^2_\mathbf{Q}) &= \prod_{i=1}^{N} N (\mathbf{q}_i | 0, \sigma^2_\mathbf{Q}),
\end{align*}
\]

where the \( \sigma^2_\mathbf{R} \), \( \sigma^2_\mathbf{P} \) and \( \sigma^2_\mathbf{Q} \) are the variances of these distributions. Let \( \Psi \) be the set of parameters, so that the joint distribution for \( \mathbf{R} \) and \( \Psi \) is

\[
    p(\mathbf{R}, \Psi) = p(\mathbf{R} | \mathbf{P}, \mathbf{Q}) p(\mathbf{P}) p(\mathbf{Q}),
\]

which denotes the CDM. Now we will describe the MDM that can explain which entry in \( \mathbf{R} \) will be missing, that is specified by the matrix \( \mathbf{X} \).

3.2. The Missing Data Model
We also obtain the MDM by using a DMF method, but it has some differences with the CDM. Since the MDM generates \( \mathbf{X} \) as a function of \( \mathbf{R} \), the rating values also determine whether the data can be observed. As such, we assume that \( \mathbf{X} \) is obtained by three factors \( \mathbf{E} \in \mathbb{R}^{M \times K} \), \( \mathbf{F} \in \mathbb{R}^{N \times K} \) and \( \mathbf{G} \in \mathbb{R}^{V \times K} \), each of them is relative to the latent matrix of users, items and rating values. In particularly, the \( v \)-th row of \( \mathbf{G} \) represents the latent vector of rating value \( v \). Similarly, each discrete entry \( x_{vi} \) in \( \mathbf{X} \) is determined by the output of another MLP network \( z_{vi} = f_{mdm}(e_v, f_i, g_v) \), where \( e_v \), \( f_i \) and \( g_v \) are the inputs of the networks. The interactions in MDM network are similar with equation (4), except the first layer is modified by

\[
    s_i = \pi_i(e_v, f_i, g_v) = \mathbf{V}_i \begin{bmatrix} e_v \\ f_i \\ g_v \end{bmatrix} + \mathbf{d}_i,
\]
where we use \( \pi_k, s_k, V_k \) and \( d_k \) to represent the transfer function, output, weight matrix and bias vector of each layer, respectively.

We also use a sigmoid function at last layer to bound the output into \((0,1)\). Since \( x_{ui} \) is a binary value, we can directly use \( u_{iv}z \) to represent the probability of \( u_{ix} = 1 \) when \( r_{ui} = v \), that is, \( p(x_{ui} = 1 | r_{ui} = v) = z_{uv} \). And we also place zero-mean Gaussian priors on \( E, F \) and \( G \). So that, the distributions of \( X \) is,

\[
p(X | E, F, G, R) = \prod_{u=1}^{M} \prod_{i=1}^{N} z_{uv}^x (1 - z_{uv})^{(1 - x_{ui})},
\]

As the same way, we let the \( \Theta \) be the set of parameters, \( \Theta = \{E, F, G\} \). So our MDM can be described by the following equation,

\[
p(X, \Theta | R) = p(X | E, F, G, R)p(E)p(F)p(G).
\]

### 3.3. The Joint Model

As we have the CDM and MDM, we can combine two models together to obtain the joint model Deep Matrix Factorization with Missing data Not At Random (DMF-MNAR). This model generates the probability that each entry \( r_{ui} \) takes a specific rating value \( v \) conditioned by the indicator \( x_{ui} \). When \( x_{ui} \) is closest to 1, it means that the entry can be observed standing a good chance. On the contrary, if \( x_{ui} \) is closest to 0, we can make a discount on the rating values. For instance, the probability of an observed entry \( r_{ui} \) taking values of \( v \) is,

\[
p^{\text{DMF-MNAR}}(r_{ui} = v | x_{ui} = 1) = p^{\text{CDM}}(r_{ui})p^{\text{MDM}}(x_{ui} = 1 | r_{ui} = v),
\]

where \( p^{\text{CDM}}(r_{ui}) \) and \( p^{\text{MDM}}(x_{ui} = 1 | r_{ui} = v) \) are the individual predictions of the CDM and MDM. In this way, the joint model improves the predictions of the CDM by using the information available in MDM.

### 4. Training Method of DMF-MNAR

As mentioned in section 2.2, a DMF method can be trained by a simple MLP network. However, our proposed model is quite different from the simple DMF method. In the first place, we can’t simply apply equation (3) as our loss function, because the CDM and MDM are probabilistic models. Moreover, our model DMF-MNAR contains two individual DMF networks to train CDM and MDM respectively. So in this section, we firstly introduce an approximation method considering the accuracy of CDM and MDM at the same time. And we proposed a pairwise neural networks for DMF-MNAR.

### 4.1. Approximate Interface

We assume that the observed data is \( R^+ \) and indicator matrix \( X \) are given, so the posterior distribution of the parameters \( \Psi, \Theta \) and missing data \( R^- \) is

\[
p^{\text{DMF-MNAR}}(\Psi, \Theta | R^+, X) = \frac{p(X, \Theta | R)p(R, \Psi)}{p(R^+, X)},
\]

where \( p(R^+, X) \) is a normalization constant. As most Bayesian models, the posterior is intractable so that we use a variational Bayesian (VB) inference to approximate [18].

Since the \( R \) and \( X \) are generated by some latent matrices \( \Psi = \{P, Q\}, \Theta = \{E, F, G\} \), we assume that \( q(\Psi) = \{q(P), q(Q)\}, q(\Theta) = \{q(E), q(F), q(G)\} \) are the variational approximations to the true posteriors of these latent matrices respectively. As we use a stochastic version of VB called SVI [15], so that our model maximizes the ELBO as following equation,
\[ L = L_{CDM} + L_{MDM}, \]

\[
L_{CDM} = E_{q(\Psi)}[\log p(R | \Psi)] - KL(q(\Psi) \| p(\Psi)),
\]

\[
L_{MDM} = E_{q(\theta)}[\log p(X | \Theta, R)] - KL(q(\Theta) \| p(\Theta)),
\]

where the \(L_{CDM}\) denotes the accuracy of CDM and \(L_{MDM}\) is for MDM. If we approximate the log likelihood with a Monte Carlo method meaning for each entry we use a single sample from posterior to estimate in every iteration, we can further decompose the equation. Because the probabilistic distribution of CDM and MDM are quite different, the expressions of ELBO are not same.

For the CDM, the ratings and latent vectors are distributed under Gaussian distributions. So that the \(L_{CDM}\) can be rewritten as,

\[
L_{CDM} = -\frac{1}{2\sigma^2_0} \sum_{u=1}^{M} \sum_{i=1}^{N} (r_{ui} - y_{ui})^2 - KL(q(P) \| p(P)) - KL(q(Q) \| p(Q))
\]

And for the MDM, the indicator \(x_{ui}\) is under a Bernoulli distribution. So that the \(L_{MDM}\) can be rewritten as,

\[
L_{MDM} = \sum_{u=1}^{M} \sum_{i=1}^{N} x_{ui} z_{ui} + (1-x_{ui})(1-z_{ui})
- KL(q(E) \| p(E)) - KL(q(F) \| p(F)) - KL(q(G) \| p(G))
\]

4.2. Neural Networks for DMF-MNAR

Figure 2. The framework of DMF-MNAR

One import aspect of deep learning is how to organize the layers in neural networks. Figure 2 illustrates the framework of our proposed model DMF-MNAR, which has two individual parts representing the CDM and MDM, respectively. As shown in the figure, the CDM and MDM both take rating matrix as the input. However, the formats of input data have small difference between two models. The CDM only take two factors represented by user and item, so that we can use a set of \((u, i, r_{ui})\) as the train data. But in the MDM, we also use the information contained by rating values, so the format of input data is \((u, i, r_{ui}, x_{ui})\). The value of \(x_{ui}\) indicates whether the entry can be observed. But in most actual situations, the data we collected are all observed, which makes training MDM intractable if we only have positive samples. So we sample items randomly from the list that user never rated before as the negative samples, where we set the rating value \(r_{ui} = 1\) and indicator \(x_{ui} = 0\). This is a widely used method for implicit feedback, but we think it’s also suitable for explicit feedback.
Because the user’s preference inferred from the discrepancy of rating values is not sufficient, according our MNAR assumption. So the negative samples will help in training the MDM, but they are redundant to the CDM.

To avoid the influences of negative samples on the CDM, we also adjust the embedding layer to suit our model. There are five latent vectors in the embedding layer, two of them belong to the CDM and the others belong to the MDM. A straightforward way is to let CDM and MDM share the same user and item latent vectors. But in that way, the negative samples will bias the predicted values. So we allow CDM and MDM to learn separate embedding layers from different input data. What’s more, this strategy can improve the flexibility of our proposed model.

Since two models are generated from different input data, it’s also hard to maximize the objective function in equation (14) at the same time. So we trained the two models alternately by their individual object functions represented in equation (15) and (16). We adopt the Adaptive Moment Estimation (Adam) in our networks. For each iteration, we firstly randomly select a small batch $D^+$ from observed data, and we train the CDM on $D^+$. After that we generated another batch $D^-$ containing the negative samples for each user in $D^+$, and we trained the MDM on $D^+ \cup D^-$. 

5. Experiments

In this section, we conduct experiments to demonstrate the performance achieved by our proposed model on various open datasets.

5.1. Experimental Settings

Datasets. We conduct experiments on three publicly accessible datasets: Yahoo! Music ratings for User Selected and Randomly Selected songs, version 1.0, MovieLens 100k and Amazon Music.

1) Yahoo! Music This dataset provides a unique opportunity to test collaborative filtering methods that incorporate missing data model [13]. This dataset consists of ratings collected during normal user interaction with Yahoo’s LaunchCast internet radio service, as well as ratings for items collected using an online survey. There are 15400 users, 1000 songs, and 311704 ratings in observed dataset. And for 5400 of these users, exactly 10 songs are randomly selected and presented to each user during the online survey. The user can listen the songs repeatedly and are enforced to rate on them. As that the dataset for randomly selected items contains 54000 ratings. The distribution of ratings for two parts are significantly different because normal interactions have more high ratings than online survey ratings. This is a strong evidence that Yahoo! dataset is under MNAR condition.

2) MovieLens This movie rating dataset has been widely used to evaluate collaborative filtering methods. We used the version containing one million ratings, where there are 6040 users, 3706 items and at least 20 ratings for each user. Although this dataset only consists of observed ratings, we randomly sample user-item interactions as the random selected ratings to train our MDM.

3) Amazon Music Amazon Music is a large dataset contains of 4.6 million reviews. As for collaborative filtering methods, we only use the rating information in the reviews. To make the dataset more appropriate for training, we filtered the dataset that only users with at least 20 interactions and items with at least 5 interactions are retained. Similar with MovieLens dataset, we also randomly sample some interactions as the random selected data.

Experimental Protocol. The experimental protocol we adopted is different from typical methods, since we deal with two rating datasets (observed dataset and random selected dataset). But our idea is very simple: we train model on observed dataset and random selected dataset and test on random selected items (Yahoo! Music) or user-selected items (MovieLens, Amazon Music). For Yahoo! Music dataset, we collect all online survey ratings as our test items. But for MovieLens and Amazon Music, the ten latest interactions for each user are hang out for test and remaining data is considered as the observed data. For each dataset, our CDM model was trained on observed data and our MDM was
trained on the combination of observed data and random selected data. The final prediction value is generated by the joint model, which mentioned in section 3.3.

**Evaluation Metrics.** We evaluate the precision of predicted rating by root-mean-square error (RMSE), which is widely used in various recommendation systems. What’s more, the output of a recommendation system usually is a ranking list where front items are more likely to be chosen. So we also evaluate the ranking performance using standard metric in information retrieval ranking applications, the normalized discounted cumulative gain (nDCG@L), where the L is the length of ranked list.

5.2. Performance Comparison

In this section, we compare our proposed model DMF-MNAR with some novel methods. As our model aims to model data under MNAR assumption by using neural networks, we mainly compare with similar methods.

- **MF-MNAR** It is a matrix factorization method learning from MNAR data [15]. The major difference between our model is that we employed neural networks based MANR assumption, which breaks the linear limitation of traditional MF methods. We adjust model according its original idea so that it can be measured by the metrics mentioned before.

- **NNMF** It is a state-of-the-art method using neural networks to train MF methods [10]. This method replaces the inner product by an arbitrary function which combines the latent vectors of user and item. However, this model is based on MAR assumption as most neural collaborative filtering methods. We implemented this method and tuned the parameters as the author mentioned.

- **DMF-MNAR** This is our proposed deep matrix factorization model based MNAR assumption. As far as we know, our model is the first to build a neural collaborative filtering method based on MNAR assumption. We implement our mode on Tensorflow, which is convenient to build neural networks. By default, we set the neg_ratio to 1, the depth of hidden layers to 3, the batch size to 25000 and the learning rate to 0.01.

The results of the comparison are summarized in Table 1. It shows that our model performed well in RMSE or nDCG in different datasets. For the Yahoo! Music dataset, our model achieves the best performance in both metrics, which can conclude that our model is more suitable for MNAR data. But for the MovieLens and Amazon Music, our model only get gains in RMSE with a close nDCG. Although our model has limitation in two datasets, we think the model is more realistic. Because the aim of recommender system is rating missing data rather than past observed data.

| Table 1: RMSE and nDCG@10 comparisons of different methods |
|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| Datasets        | Metrics | MF-MNAR | NNMF | DMF-MNAR |
|----------------|----------|---------|------|----------|
| Yahoo Music!     | RMSE     | 1.4875  | 1.0853 | **1.0536** |
|                 | nDCG@10  | 0.8209  | 0.8167 | **0.8224** |
| MovieLens        | RMSE     | 1.043   | 1.0465 | **1.0313** |
|                 | nDCG@10  | 0.889   | 0.8901 | 0.8611   |
| Amazon Music     | RMSE     | 1.2101  | 1.0839 | **1.0405** |
|                 | nDCG@10  | 0.8826  | **0.9136** | 0.8965   |

5.3. Sensitivity to Hyper-Parameters

We also explore the performance of our proposed model under different parameters settings.

**Depth of Hidden Layers** In the neural network of our proposed model, we used MLP to map the users and items to latent vectors. The depth of hidden layers H determines the model’s capability, and we conduct some elaborate experiments to find effective depth. Figure 3 illustrates the performance of different layers while training on Yahoo! Music. As shown in the results, a single layer gets the worst
performance because of lesser learning ability. And the 2-layer and 3-layer have similar results, but it seems that 3-layer converge to stable state in less iterations. We also tried deeper layers, but it’s not useful to produce additional gains. And the higher depth will make it more time consuming to train the model. So the best depth of layers we suggested is 2 or 3 according the experiments.

![Graph showing performance of DMF-MNAR with different layers](image)

**Figure 3.** The performances of DMF-MNAR with different layers

**Size of Latent Vectors** Same as the other MF methods, the size of latent vectors in embedding layer is possibly a sensitive parameter in our model. Usually a small vector will limit the generalization ability of model, but a large vector may reduce the accuracy of model. So we also conduct some experiments with different sizes of latent vectors on Yahoo! Music to explore suitable parameters setting. As shown in Table 2, the model with 16-size latent vectors performs best compared with other parameters. So our experimental result also proves that small or large vectors are ineffective to our model, which similar with other MF methods.

| Metrics     | 8   | 16  | 32  | 64  |
|-------------|-----|-----|-----|-----|
| RMSE        | 1.0535 | **1.0553** | 1.0678 | 1.2898 |
| nDCG@10     | 0.8060 | **0.8128** | 0.8108 | 0.8001 |

**Table 2. Results of different K on Yahoo! Music**

6. Conclusion and Future Work

In this work, we study the problem of learning collaborative filtering method based MNAR assumption. Our proposed model is the first practical implementation to use deep matrix factorization for ordinal matrix data with entries missing not at random. In our model, there are two parts representing the CDM model and the MDM model, respectively. And we combined two models together as the joint model to describe the whole MNAR data. We also proposed a method using variational bayesian inference to approximate the distribution of our model. In the other hand, we designed a pairwise neural networks to train the model, which used generation ability of deep learning. The experiments on several datasets demonstrate that our proposed model produces gains in various metrics.

Up to now, we only use the rating matrix as our input data, and the latent features are generated by identity of user and item, which may limit the expression of those features. In the future, we will use extra data, such as social relation, review text and history data, to model features of user and item. What’s more, to make the recommender system more intelligent, we will consider to employ reinforcement learning in our model. Therefore, we can evaluate a long-term reward while users are interacting with recommender system. We believe that the mechanism of deep learning can improve the recommender system.
References
[1] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.
[2] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Largescale parallel collaborative filtering for the netflix prize. *Lecture Notes in Computer Science*, 5034:337–348, 2008.
[3] 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.
[4] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.
[5] 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.
[6] Robin Devooght, Nicolas Kourtellis, and Amin Mantrach. Dynamic matrix factorization with priors on unknown values. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 189–198. ACM, 2015.
[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016.
[8] 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.
[9] Florian Strub and J’er’emie Mary. Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *NIPS workshop on machine learning for eCommerce*, 2015.
[10] Gintare Karolina Dziugaite and Daniel M Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.
[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.
[12] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2014.
[13] Benjamin M Marlin and Richard S Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*, pages 5–12. ACM, 2009.
[14] Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–722. ACM, 2010.
[15] Jose Miguel Hernandez-Lobato, Neil Houlsby, and Zoubin Ghahramani. Probabilistic matrix factorization with non-random missing data. In *International Conference on Machine Learning*, pages 1512–1520, 2014.
[16] Tao Chen, Xiangnan He, and Min-Yen Kan. Context-aware image tweet modelling and recommendation. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 1018–1027. ACM, 2016.
[17] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.
[18] Manfred Opper and David Saad. *Advanced mean field methods: Theory and practice*. MIT press, 2001.