From Personalization to Privatization: Meta Matrix Factorization for Private Rating Predictions

Lin, Y.; Ren, P.; Chen, Z.; Yu, D.; Ma, J.; de Rijke, M.; Cheng, X.

Publication date
2019

Document Version
Submitted manuscript

License
Unspecified

Link to publication

Citation for published version (APA):
Lin, Y., Ren, P., Chen, Z., Yu, D., Ma, J., de Rijke, M., & Cheng, X. (2019). *From Personalization to Privatization: Meta Matrix Factorization for Private Rating Predictions*. (v1 ed.) ArXiv.

From Personalization to Privatization: Meta Matrix Factorization for Private Rating Predictions

Yujie Lin[1], Pengjie Ren[2], Zhumin Chen[1], Zhaochun Ren[1], Dongxiao Yu[1], Jun Ma[1], Maarten de Rijke[2], Xiuzhen Cheng[1]
[1] Shandong University, Qingdao, China
[2] University of Amsterdam, Amsterdam, The Netherlands
yu.jie.lin@outlook.com, p.ren@uva.nl, {chenzhumin, zhaochun.ren, dxyu, majun}@sdu.edu.cn, derijke@uva.nl, xzcheng@sdu.edu.cn

Abstract

Matrix factorization (MF) techniques have been shown to be effective for rating predictions (RPs) in personalized recommender systems. Existing MF methods use the same item embeddings and the same RP model for all users, while ignoring the possibility that different users may have different views about the same item and may favor different RP models. We introduce a novel MF framework, named meta matrix factorization (MetaMF), that generates private item embeddings and RP models. Given a vector representing a user, we first obtain a collaborative vector by collecting useful information from all users with a collaborative memory (CM) module. Then, we employ a meta recommender (MR) module to generate private item embeddings and a RP model based on the collaborative vector. To address the challenge of generating a large number of high-dimensional item embeddings, we devise a rise-dimensional generation (RG) strategy that first generates a low-dimensional item embedding matrix and a rise-dimensional matrix, and then multiply them to obtain high-dimensional embeddings. Finally, we use the generated model to produce private RPs for a given user. Experiments on two benchmark datasets show that MetaMF outperforms state-of-the-art MF methods. MetaMF generates similar/dissimilar item embeddings and models for different users to flexibly exploit collaborative filtering (CF), demonstrating the benefits of MetaMF.

1 Introduction

Rating predictions (RP) have been studied for decades as a branch of research in recommender systems (Marlin 2004; Koren 2008; Li and She 2017). Unlike ranking predictions, the goal of RPs is to predict the rating that a user would give to an item that she has not rated in the past as precisely as possible (Koren, Bell, and Volinsky 2009; Hu, Sun, and Liu 2014). This is useful not only for recommendation purposes but also when there is a need to estimate users' opinions about a particular item (Li et al. 2017).

The general idea of matrix factorization (MF) is to optimize latent factors to represent users and items by projecting users and items into a joint dense vector space (Koren, Bell, and Volinsky 2009; He et al. 2017). Conventional MF methods, such as singular value decomposition (SVD) (Koren 2008), probabilistic matrix factorization (PMF) (Mnih and Salakhutdinov 2008) and non-negative matrix factorization (NMF) (Zhang et al. 2006), decompose the rating matrix into a user factor matrix and a shared item factor matrix for all users. Deep learning based MF methods further take the linear and non-linear interactions between users and items into consideration by employing restricted Boltzmann machine (RBM) (Salakhutdinov, Mnih, and Hinton 2007), autoencoder (AE) (Sedhain et al. 2015; Strub, Gaudel, and Mary 2016), convolutional neural network (CNN) (Kim et al. 2016) or multi-layer perceptron (MLP) (He et al. 2017; Xue et al. 2017). Recently, various methods have been proposed to enhance MF by incorporating side information (Wang et al. 2018; Cheng et al. 2018b; 2018a; Chen and de Rijke 2018; Xiao et al. 2019a; 2019b).

All MF methods use the same RP model as well as item embeddings for all users to predict personalized ratings. We hypothesize that this is not always optimal. First, different users might have different views and/or angles about the same item, which indicates they should not always share the same item embeddings. For example, each reader has her own unique understanding of "Hamlet." Second, different users might favor different RP strategies, which means they should not consistently use the same RP model. Consider

Table 1: Performance of different RP models for three users from the Hetrec2011-movielens dataset.

Method	User 4		User 90		User 1983	
	MAE	MSE	MAE	MSE	MAE	MSE
PMF	**0.280**	**0.125**	0.727	0.770	**0.333**	0.189
NMF	1.040	1.630	**0.449**	**0.332**	0.398	**0.168**

Table 1, where we select three users from the test set of a benchmark dataset and list the RP performance of two competitive MF methods. PMF is more suitable for user 4, but performs very badly for user 90. In contrast, NMF is good for user 90, but is not suitable for user 4. This is because there are different factors that should be considered for different users for RPs and it is hard for a single RP model to perfectly capture all factors.

We propose to adopt private RPs, where each user has her own item embeddings and RP model. A key challenge is how to build private RP models and at the same time effectively utilize CF information due to the fact that we may not have enough personal data for each user to build her own model. Besides, it is also unrealistic to store and maintain a separate model for each individual user. In this paper, we address this by introducing a novel matrix factorization framework, namely *meta matrix factorization* (MetaMF). Instead of building a model for each user, we propose to "generate" private item embeddings and RP models with a MetaMF model. Specifically, we assign a so-called indicator vector (i.e., a one-hot vector corresponding to a user id) to each user. For a given user, we first fuse her indicator vector to get a collaborative vector by collecting useful information from other users with a collaborative memory (CM) module. Then, we employ a meta recommender (MR) module to generate private item embeddings and a RP model based on the collaborative vector. It is challenging to directly generate the item embeddings due to the large number of items and the high dimensions. To tackle this, we devise a rise-dimensional generation (RG) strategy that first generates a low-dimensional item embedding matrix and a rise-dimensional matrix, and then multiply them to obtain high-dimensional embeddings. Finally, we use the generated model to obtain private RPs for this user.

We perform extensive experiments on two benchmark datasets. MetaMF outperforms state-of-the-art MF methods. Both the generated item embeddings and the RP model parameters exhibit clustering phenomena, demonstrating that MetaMF can effectively model CF while generating a private model for each user.

The main contributions of this paper are as follows:

- We introduce a novel MetaMF framework for rating predictions, which is the first to realize private rating predictions, to the best of our knowledge.
- We devise collaborative memory and meta recommender modules as well as a rise-dimensional generation strategy to implement MetaMF.
- We conduct experiments and analyses on two datasets to verify the effectiveness of MetaMF.

2 Related Work

2.1 Matrix Factorization

Matrix factorization (MF) has attracted a lot attention since it was proposed for recommendation. Early studies focus mainly on how to achieve better rating matrix decomposition. Sarwar et al. (2000) employ SVD to reduce the dimensionality of the rating matrix, so that they can get low-dimensional user and item vectors. Goldberg et al. (2001) apply principal component analysis (PCA) to decompose the rating matrix, and obtain the principle components as user or item vectors. Zhang et al. (2006) propose NMF which decomposes the rating matrix by modeling each user's ratings as an additive mixture of rating profiles from user communities or interest groups and constraining the factorization to have non-negative entries. Mnih and Salakhutdinov (2008) propose PMF to model the distributions of user and item vectors from a probabilistic point of view. Koren (2008) proposes SVD++, which enhances SVD by including implicit feedback as opposed to SVD, which only includes explicit feedback.

The matrix decomposition methods mentioned above estimate ratings by simply calculating the inner product between the user and item vectors, which is not sufficient to capture their complex interactions. Deep learning has been introduced to MF for better modeling of the user-item interactions with non-linear transformations. Sedhain et al. (2015) propose AutoRec, which takes ratings as input and reconstructs the ratings by an autoencoder. Later, Strub, Gaudel, and Mary (2016) enhance AutoRec by incorporating side information into a denoising autoencoder. He et al. (2017) propose the neural collaborative filtering (NCF), which employs MLP to model the user-item interactions. Xue et al. (2017) present the deep matrix factorization (DMF) which enhances NCF by considering both explicit and implicit feedback. He et al. (2018) use CNNs to improve NCF and present the ConvNCF which uses the outer product to model user-item interactions. Cheng et al. (2018a) introduce an attention mechanism into NCF to differentiate the importance of different user-item interactions. Recently, a number of studies have investigated the use of side information or implicit feedback to enhance these neural models (Li and She 2017; Xiao et al. 2019a; 2019b; Yi et al. 2019).

All these models provide personalized RPs by learning user representations to encode the differences among users, while sharing item embeddings and models. In contrast, MetaMF provides "private" RPs by generating non-shared models as well as item embeddings for different users.

2.2 Meta Learning

Meta learning, also known as "learning to learn," has shown its effectiveness in reinforcement learning (Xu, van Hasselt, and Silver 2018), few-shot learning (Nichol, Achiam, and Schulman 2018), image classification (Ravi and Larochelle 2017). Below, we survey the most closely related works.

Some meta learning works aim to learn a special network used to generate the parameters of other networks. Jia et al. (2016) propose a network to dynamically generate filters for CNNs. Bertinetto et al. (2016) introduce a model to predict the parameters of a pupil network from a single exemplar for one-shot learning. Ha, Dai, and Le (2016) propose hypernetworks, which employ a network to generate the weights of another network. Krueger et al. (2017) present a Bayesian variant of hypernetworks that learns the distribution over the parameters of another network. Chen et al. (2018b) use a hypernetwork to share function-level information across multiple tasks. However, none of them targets recommendation which is a more complex task with its own new challenges.

Recently, some studies try to introduce meta learning into recommendations. Vartak et al. (2017) study the item cold-start problem in recommendations from a meta-learning perspective. They view recommendation as a binary classification problem, where the class labels indicate whether the user engaged with the item. Then they devise a classifier by adapting a few-shot learning paradigm (Snell, Swersky, and Zemel 2017). Chen et al. (2018a) propose a recommen-

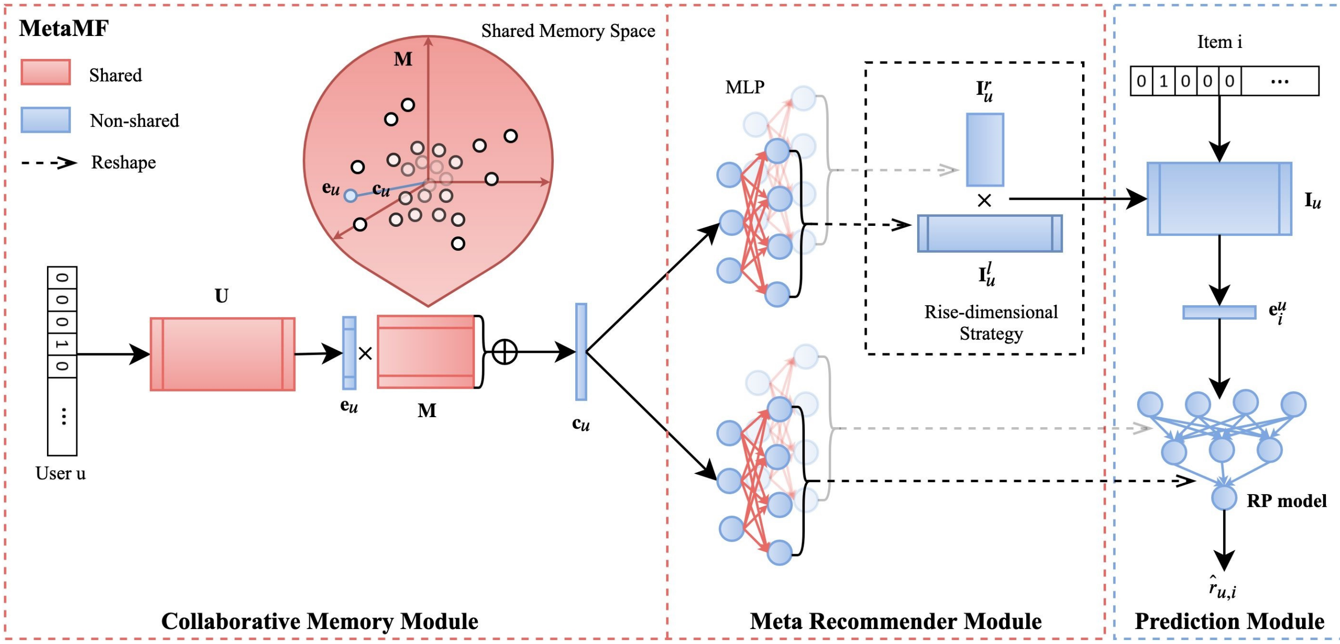


Figure 1: An overview of MetaMF. It consists of three modules. The collaborative memory module and the meta recommender module with the rise-dimensional strategy are shared to generate private item embeddings and RP models for different users. The prediction module is non-shared which predicts private ratings based on the generated item embeddings and RP models for each user.

dation framework based on federated meta learning, which maintains a shared model in the cloud. To adapt it for each user, they download the model to the local device and fine-tune the model for personalized recommendations. Different from these publications, we learn a hypernetwork (i.e., MetaMF) to directly generate private MF models for each user for RPs.

3 Meta Matrix Factorization

3.1 Overview

Given a user u and an item i, the goal of rating prediction is to estimate a rating $\hat{r}_{u,i}$ that is as accurate as the true rating $r_{u,i}$. We denote the user set as $\mathcal{U}$, the item set as $\mathcal{I}$, the true rating set as $\mathcal{R}$, which will be divided into the training set D_{train}, the valid set D_{valid}, and the test set D_{test}.

As shown in Fig. 1, MetaMF contains three modules: a *collaborative memory module*, a *meta recommender module* and a *prediction module*, where the collaborative memory module and the meta recommender module are shared by all users, and the prediction module is non-shared. In the collaborative memory module, we first obtain the user embedding $\mathbf{e}_u$ of u from the user embedding matrix $\mathbf{U}$ and take it as the coordinates to obtain the collaborative vector $\mathbf{c}_u$ from a shared memory space that fuses the information from all users. Then we input $\mathbf{c}_u$ to the meta recommender module to generate the parameters of a private RP model for u. The RP model can be of any type. In this work, the RP model is a multi-layer perceptron (MLP). We also generate the private item embedding matrix $\mathbf{I}_u$ of u with a *rise-dimensional generation strategy*. Finally, the prediction module takes the item embedding $\mathbf{e}_{ui}$ of i from $\mathbf{I}_u$ as input and predicts $r_{u,i}$ using the generated RP model.

Next we detail each module.

3.2 Collaborative Memory Module

In order to facilitate collaborative filtering, we propose the CM module to learn a collaborative vector for each user, which encodes both the user's own information and some useful information from the other users.

Specifically, we assign each user u and each item i the indicator vectors, $\mathbf{i}_u \in \mathbb{R}^m$ and $\mathbf{i}_i \in \mathbb{R}^n$, where m is the number of users and n is the number of items. Note that $\mathbf{i}_u$ and $\mathbf{i}_i$ are one-hot vectors with each dimension corresponding to a particular user or item. For the given user u, we first get the user embedding $\mathbf{e}_u$ by Eq. (1):

$$\mathbf{e}_u = \mathbf{U}\mathbf{i}_u, \tag{1}$$

where $\mathbf{e}_u \in \mathbb{R}^{d_u}$, $\mathbf{U} \in \mathbb{R}^{d_u \times m}$ is the user embedding matrix, and d_u is the size of user embeddings. Then we proceed to get the collaborative vector for u. Specifically we use a shared memory matrix $\mathbf{M} \in \mathbb{R}^{d_u \times k}$ to store the basis vectors which span a space of all collaborative vectors, where k is the dimension of basis vectors and collaborative vectors. And we consider the user embedding $\mathbf{e}_u$ as the coordinates of u in the shared memory space. So the collaborative vector $\mathbf{c}_u \in \mathbb{R}^k$ for u is the linear combination of the basis vectors in $\mathbf{M}$ by $\mathbf{e}_u$, as shown in Eq. (2):

$$\mathbf{c}_u = \sum_i \mathbf{M}(i,:)\mathbf{e}_u(i), \tag{2}$$

where $\mathbf{M}(i,:)$ is the i-th vector of $\mathbf{M}$ and $\mathbf{e}_u(i)$ is the i-th scalar of $\mathbf{e}_u$. Because the memory matrix $\mathbf{M}$ is shared among all users, the shared memory space will fuse the information from all users. MetaMF can flexibly exploit collaborative filtering among users by assigning them with similar collaborative vectors in the space defined by $\mathbf{M}$, which is equivalent to learning similar user embeddings as in existing MF methods.

3.3 Meta Recommender Module

We propose the MR module to generate the private item embeddings and RP model based on the collaborative vector from the CM module.

Private Item Embeddings. We propose to generate the private item embedding matrix $\mathbf{I}_u \in \mathbb{R}^{d_i \times n}$ for each user u, where d_i is the size of item embeddings. However, it is unrealistic to directly generate the whole item embedding matrix because there are usually a large number of items (i.e., n) and their embeddings are high-dimensional (i.e., d_i). Therefore, we propose the rise-dimensional generation (RG) strategy to decompose the generation into two parts: a low-dimensional item embedding matrix $\mathbf{I}_u^l \in \mathbb{R}^{s \times n}$ and a rise-dimensional matrix $\mathbf{I}_u^r \in \mathbb{R}^{d_i \times s}$, where s is the size of low-dimensional item embeddings and $s \ll d_i$. Specifically, we first follow Eq. (3) to generate $\mathbf{I}_u^l \in \mathbb{R}^{sn}$ and $\mathbf{I}_u^r \in \mathbb{R}^{d_i s}$ (in the form of vectors):

$$
\begin{aligned}
\mathbf{h}_i^l &= \mathrm{ReLU}(\mathbf{W}_i^l \mathbf{c}_u + \mathbf{b}_i^l) \\
\mathbf{I}_u^l &= \mathbf{U}_i^l \mathbf{h}_i^l \\
\mathbf{h}_i^r &= \mathrm{ReLU}(\mathbf{W}_i^r \mathbf{c}_u + \mathbf{b}_i^r) \\
\mathbf{I}_u^r &= \mathbf{U}_i^r \mathbf{h}_i^r,
\end{aligned}
\tag{3}
$$

where $\mathbf{W}_i^l$ and $\mathbf{W}_i^r \in \mathbb{R}^{o \times k}$, $\mathbf{U}_i^l \in \mathbb{R}^{sn \times o}$ and $\mathbf{U}_i^r \in \mathbb{R}^{d_i s \times o}$ are weights; $\mathbf{b}_i^l$ and $\mathbf{b}_i^r \in \mathbb{R}^o$ are biases; $\mathbf{h}_i^l$ and $\mathbf{h}_i^r \in \mathbb{R}^o$ are hidden states; o is the hidden size. Then we reshape $\mathbf{I}_u^l$ to a matrix whose shape is $s \times n$, and reshape $\mathbf{I}_u^r$ to a matrix whose shape is $d_i \times s$. Finally, we multiply $\mathbf{I}_u^l$ and $\mathbf{I}_u^r$ to get $\mathbf{I}_u$:

$$
\mathbf{I}_u = \mathbf{I}_u^r \mathbf{I}_u^l. \tag{4}
$$

For different users, the generated item embedding matrices are different.

Private RP Model. We also propose to generate a private RP model for each user u. We use a MLP as the RP model, so we need to generate the weights and biases for each layer of MLP. Specifically, for layer l, we denote its weights and biases as $\mathbf{W}_l^u \in \mathbb{R}^{f_{out} \times f_{in}}$ and $\mathbf{b}_l^u \in \mathbb{R}^{f_{out}}$ respectively, where f_{in} is the size of its input and f_{out} is the size of its output. Then $\mathbf{W}_l^u$ and $\mathbf{b}_l^u$ are calculated as follows:

$$
\begin{aligned}
\mathbf{h}_g &= \mathrm{ReLU}(\mathbf{W}_g^h \mathbf{c}_u + \mathbf{b}_g^h) \\
\mathbf{W}_l^u &= \mathbf{U}_g^w \mathbf{h}_g + \mathbf{b}_g^w \\
\mathbf{b}_l^u &= \mathbf{U}_g^b \mathbf{h}_g + \mathbf{b}_g^b,
\end{aligned}
\tag{5}
$$

where $\mathbf{W}_g^h \in \mathbb{R}^{o \times k}$, $\mathbf{U}_g^w \in \mathbb{R}^{f_{out} f_{in} \times o}$ and $\mathbf{U}_g^b \in \mathbb{R}^{f_{out} \times o}$ are weights; $\mathbf{b}_g^h \in \mathbb{R}^o$, $\mathbf{b}_g^w \in \mathbb{R}^{f_{out} f_{in}}$ and $\mathbf{b}_g^b \in \mathbb{R}^{f_{out}}$ are biases; $\mathbf{h}_g \in \mathbb{R}^o$ is hidden state. Finally, we reshape $\mathbf{W}_l^u$ to a matrix whose shape is $f_{out} \times f_{in}$. Note that $\mathbf{W}_g^h$, $\mathbf{b}_g^h$, $\mathbf{U}_g^w$, $\mathbf{b}_g^w$, $\mathbf{U}_g^b$ and $\mathbf{b}_g^b$ are not shared by different layers of the RP model. And f_{in} and f_{out} also vary with different layers. Detailed settings can be found in the experimental setup. Also, MetaMF returns different parameters of the MLP to each user.

3.4 Prediction Module

The prediction module estimates the user's rating for a given item i using the generated item embedding matrix $\mathbf{I}_u$ and RP model from the CM module.

First, we get the private item embedding $\mathbf{e}_i^u \in \mathbb{R}^{d_i}$ of i from $\mathbf{I}_u$ by Eq. (6):

$$
\mathbf{e}_i^u = \mathbf{I}_u \mathbf{i}_i. \tag{6}
$$

Then we follow Eq. (7) to predict $r_{u,i}$ based on the RP model:

$$
\begin{aligned}
\mathbf{h}_1 &= \mathrm{ReLU}(\mathbf{W}_1^u \mathbf{e}_i^u + \mathbf{b}_1^u) \\
\mathbf{h}_2 &= \mathrm{ReLU}(\mathbf{W}_2^u \mathbf{h}_1 + \mathbf{b}_2^u) \\
&\vdots \\
\mathbf{h}_{L-1} &= \mathrm{ReLU}(\mathbf{W}_{L-1}^u \mathbf{h}_{L-2} + \mathbf{b}_{L-1}^u) \\
\hat{r}_{u,i} &= \mathbf{W}_L^u \mathbf{h}_{L-1} + \mathbf{b}_L^u,
\end{aligned}
\tag{7}
$$

where L is the number of layers of the RP model. The weights $\{\mathbf{W}_1^u, \mathbf{W}_2^u, \ldots, \mathbf{W}_{L-1}^u, \mathbf{W}_L^u\}$ and biases $\{\mathbf{b}_1^u, \mathbf{b}_2^u, \ldots, \mathbf{b}_{L-1}^u, \mathbf{b}_L^u\}$ are generated by the CM module. The last layer L is the output layer which returns a scalar as the predicted rating $\hat{r}_{u,i}$.

3.5 Loss

In order to learn MetaMF, we formulate the RP task as a regression problem and the loss function is defined as:

$$
L_{rp} = \frac{1}{|D_{train}|} \sum_{r_{u,i} \in D_{train}} (r_{u,i} - \hat{r}_{u,i})^2. \tag{8}
$$

To avoid overfitting, we add the L2 regularization term:

$$
L_{reg} = \frac{1}{2} \|\Theta\|_2^2, \tag{9}
$$

where Θ represents the trainable parameters of MetaMF. Note that unlike existing MF methods, the item embeddings and the parameters of RP models are not included in Θ, because they are also the outputs of MetaMF, not trainable parameters.

The final loss L is a linear combination of L_{rp} and L_{reg}:

$$
L = L_{rp} + \lambda L_{reg}, \tag{10}
$$

where λ is the weight of L_{reg}. The whole framework of MetaMF can be efficiently trained using back-propagation in an end-to-end paradigm.

4 Experimental Setup

4.1 Datasets

We conduct experiments on two widely used datasets: **Douban** (Hu, Sun, and Liu 2014) and **Hetrec2011-movielens** (Cantador, Brusilovsky, and Kuflik 2011). We list the statistics of these two datasets in Table 2. For each dataset, we randomly separate it into three chunks: 80% as the training set, 10% as the validation set and 10% as the test set.

Table 2: Statistics of the datasets, where *#avg* means the average rating number of each user.

Datasets	***#users***	***#items***	***#ratings***	***#avg***
Douban	2,509	39,576	894,887	357
Hetrec2011-movielens	2,113	10,109	855,599	405

4.2 Baselines

We compare MetaMF with the following conventional and deep learning-based MF methods. It is worth noting that in this paper we focus on predicting ratings based on rating matrices, thus for fairness we neglect the MF methods which need side information.

- **Conventional methods:**
  - **URP** (Marlin 2004): It employs a topic model to model user preference.
  - **NMF** (Zhang et al. 2006): It uses non-negative matrix factorization to decompose rating matrices.
  - **PMF** (Mnih and Salakhutdinov 2008): It applies Gaussian distributions to model the latent factors of users and items.
  - **SVD++** (Koren 2008): It extends SVD by considering implicit feedback for modeling latent factors.
  - **LLORMA** (Lee et al. 2016): It uses a number of low-rank submatrices to compose rating matrices.
- **Deep learning-based methods:**
  - **RBM** (Salakhutdinov, Mnih, and Hinton 2007): It employs RBM to model the generation process of ratings.
  - **AutoRec** (Sedhain et al. 2015): It proposes AEs to model the interactions between users and items. AutoRec has two variants, one taking users' ratings as input, denoted as AutoRec-U, and the other taking items' ratings as input, denoted as AutoRec-I.
  - **CFN** (Strub, Gaudel, and Mary 2016): It enhances AutoRec by introducing a denoising autoencoder. CFN also has two variants, called CFN-U and CFN-I.
  - **NCF** (He et al. 2017): This is the state-of-the-art MF method that combines generalized matrix factorization and MLP to model user-item interactions. We adapt NCF for the RP task by dropping the sigmoid activation function on its output layer and replacing its loss function with Eq. (8).

4.3 Implementation Details

The user embedding size d_u and the item embedding size d_i are set to 16. The size of the collaborative vector k is set to 128. The size of the low-dimensional item embedding s is set to 8. The hidden size o is set to 512. And the RP model in the prediction module is an MLP with two layers whose layer sizes are 8 and 1. During training, we initialize all trainable parameters randomly with the Xavier method (Glorot and Bengio 2010). We choose Adam (Kingma and Ba 2014) to optimize MetaMF, set the learning rate to 0.0001, and set the regularizer weight λ to 0.001. We use a mini-batch size 64 by grid search. Our framework is implemented with Pytorch. In our experiments, we implement NCF based on the released code of the author.[1] We refer the release code[2] to realize AutoRec and CFN. And we use LibRec[3] to implement the other baselines.

4.4 Evaluation Metrics

To evaluate the performance of rating prediction methods, we employ two evaluation metrics, i.e., Mean Absolute Error (MAE) and Mean Square Error (MSE). Both of them are widely applied for the RP task in recommender systems. Given the predicted rating $\hat{r}_{u,i}$ and the true rating $r_{u,i}$ of user u on item i in the test set D_{test}, MAE is calculated as:

$$\text{MAE} = \frac{1}{|D_{test}|} \sum_{r_{u,i} \in D_{test}} |r_{u,i} - \hat{r}_{u,i}|. \tag{11}$$

Whereas MSE is defined as:

$$\text{MSE} = \frac{1}{|D_{test}|} \sum_{r_{u,i} \in D_{test}} (r_{u,i} - \hat{r}_{u,i})^2. \tag{12}$$

In our experiments, statistical significance is tested using a two-sided paired t-test for significant differences ($p < 0.05$).

5 Experimental Results

5.1 Research Questions

We seek to answer the following research questions in our experiments:

(**RQ1**) Does the proposed MetaMF method outperform the state-of-the-art MF methods on the rating prediction task?

(**RQ2**) Does generating private item embeddings improve the performance of rating predictions?

(**RQ3**) Is generating private RP models helpful to make rating predictions better?

(**RQ4**) Can MetaMF generate different item embeddings and RP models for different users while exploiting collaborative filtering?

5.2 Performance Comparison (RQ1)

We start by addressing RQ1 and test if MetaMF outperforms the state-of-the-art MF methods. Table 3 lists the rating prediction performance of all MF methods. Our main observations are as follows:

(1) MetaMF outperforms other baselines in terms of all metrics on all datasets. For the Douban dataset, MetaMF achieves a significant 0.008 (0.011) decrease over NCF in terms of MAE (MSE); and on the Hetrec2011-movielens dataset, it achieves a 0.017 (0.034) decrease over NCF in terms of MAE (MSE). There are three reasons to explain these results. Firstly, MetaMF generates private item embeddings for different users, which

[1]https://github.com/hexiangnan/neural_collaborative_filtering

[2]https://github.com/gtshs2/Autorec

[3]https://www.librec.net/

Table 3: Comparison results of MetaMF and baselines on the two datasets.

Method	Douban		Hetrec2011-movielens	
	MAE	MSE	MAE	MSE
URP	0.762	0.865	0.787	1.006
NMF	0.602	0.585	0.625	0.676
PMF	0.639	0.701	0.617	0.644
SVD++	0.593	0.570	0.579	0.590
LLORMA	0.610	0.623	0.588	0.603
RBM	1.058	1.749	1.124	1.947
AutoRec-U	0.709	0.911	0.660	0.745
AutoRec-I	0.704	0.804	0.633	0.694
CFN-U	0.707	0.907	0.659	0.743
CFN-I	0.634	0.646	0.597	0.619
NCF	0.594	0.564	0.590	0.614
MetaMF	**0.586**†	**0.553**	**0.573**†	**0.580**†

Bold face indicates leading results in terms of the corresponding metric; † indicates that MetaMF significantly outperforms NCF.

can capture the differences among users' views and/or angles on the same item. Secondly, MetaMF provides different users with private RP models, which allows MetaMF to better model the user's profiles. Lastly, MetaMF can take advantage of collaborative filtering through the collaborative memory module, so users can share information as in ordinary MF methods.

(2) The item embedding size used in MetaMF is half that of NCF,[4] and the MLP used in the prediction module is also simpler than the one in NCF.[5] However, MetaMF still outperforms NCF. This indicates that since each user has her own item embeddings (and RP model), we can reduce their sizes (scale) while still achieving competitive RP performance. We also tried larger embedding sizes (model scale), but in that case the performance MetaMF slightly drops due to overfitting.

(3) Although conventional methods cannot model non-linear transformations as well as deep learning-based methods, we see they still show comparable performance. In Table 3, NMF, PMF and LLORMA outperform RBM, AutoRec and CFN-U. There may be two reasons. On one hand, RBM, AutoRec and CFN do not explicitly model user latent factors and item latent factors, which hinders them from learning better user and item representations. On the other hand, we guess that the linear models may be more suitable to some users. Accordingly, we conclude that deep learning-based models are not the best choices for all users, which also supports our argument that we should provide private RP models for users.

(4) SVD++ performs well on both datasets, and outperforms NCF on the Hetrec2011-movielens dataset. The reason is because that SVD++ considers implicit feedback, which reflects interactions between a given item and another item that the user rates. Thus, compared to other baselines, SVD++ can better capture personalized factors in the rating prediction task. However MetaMF is also better than SVD++, since MetaMF can better model users' private behaviors or views.

(5) CFN achieves a better performance than AutoRec. The denoising autoencoder can improve the robustness of models. And AutoRec-I and CFN-I outperform AutoRec-U and CFN-U, respectively. Because the number of items is bigger than the number of users, reconstructing item ratings is easier than reconstructing user ratings.

[4] We set the item embedding size to 32 for NCF

[5] In experiments, NCF has four layers with sizes of $[32, 16, 8, 1]$ respectively

Table 4: Comparison results of MetaMF and MetaMF-SI on the two datasets.

Method	Douban		Hetrec2011-movielens	
	MAE	MSE	MAE	MSE
MetaMF-SI	0.587	0.554	0.591	0.616
MetaMF	**0.586**	**0.553**	**0.573**	**0.580**

Table 5: Comparison results of MetaMF and MetaMF-SM on the two datasets.

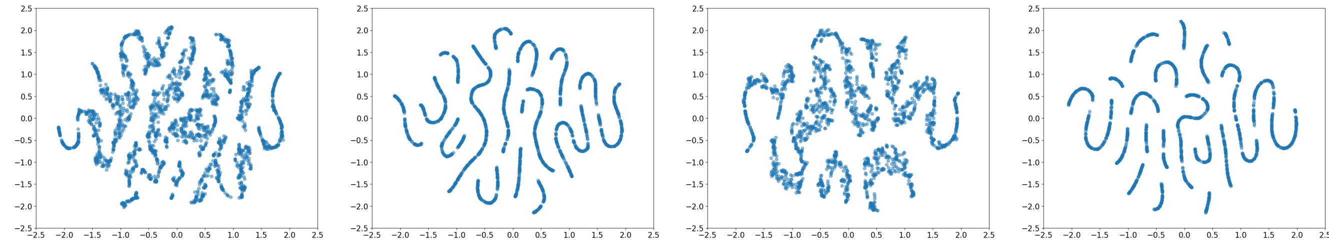
Method	Douban		Hetrec2011-movielens	
	MAE	MSE	MAE	MSE
MetaMF-SM	0.599	0.577	0.588	0.609
MetaMF	**0.586**	**0.553**	**0.573**	**0.580**

5.3 Effectiveness of Generating Private Item Embeddings (RQ2)

Next we address RQ2 to analyze the effectiveness of generating private item embeddings for the RP task. We compare MetaMF with MetaMF-SI which only generates private RP models for different users while sharing a common item embedding matrix among all users. As shown in Table 4, MetaMF outperforms MetaMF-SI on the Hetrec2011-movielens dataset. We conclude that generating private item embeddings for each user can improve the performance of RPs. As each user has her own perspective, generating a specific item embedding for each user pays off. Possibly because users in the Douban dataset have similar views or angles, we notice that MetaMF-SI and MetaMF have comparable performance on the Douban dataset.

5.4 Effectiveness of Generating Private RP Models (RQ3)

To help us answer RQ3, we compare MetaMF with MetaMF-SM, which generates different item embeddings for different users and shares a common RP model among all users. From Table 5, we can see that MetaMF consistently outperforms MetaMF-SM on both datasets. Thus, generating private RP models for users is able to improve the performance of RPs. Because different users take different ways

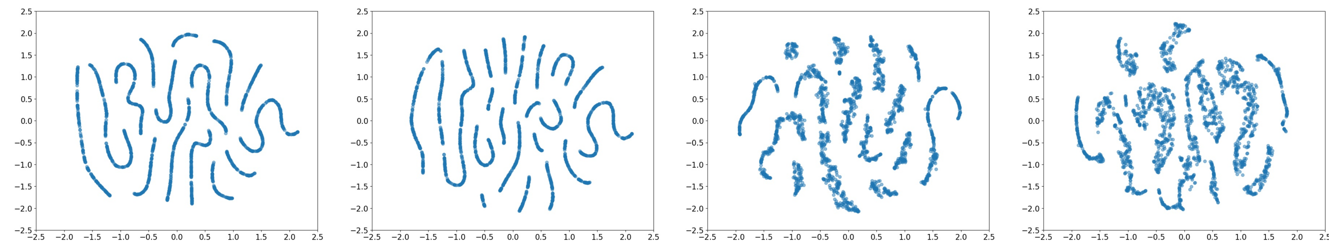


(a) The weights of layer 1 on the Douban dataset.

(b) The weights of layer 2 on the Douban dataset.

(c) The weights of layer 1 on the Hetrec2011-movielens dataset.

(d) The weights of layer 2 on the Hetrec2011-movielens dataset.



(e) The embeddings of item 4135 on the Douban dataset.

(f) The embeddings of item 24878 on the Douban dataset.

(g) The embeddings of item 2201 on the Hetrec2011-movielens dataset.

(h) The embeddings of item 9325 on the Hetrec2011-movielens dataset.

Figure 2: The generated weights and item embeddings reduced dimension by t-SNE and normalized by mean and standard deviance on the two datasets, where one point corresponds to one user.

to interact with items, a shared RP model is not suitable for all users.

Furthermore, by comparing MetaMF-SI and MetaMF-SM, we see that MetaMF-SI outperforms MetaMF-SM on the Douban dataset, but MetaMF-SM outperforms MetaMF-SI on the Hetrec2011-movielens dataset. Users of the Douban dataset are prone to interact with items in different ways, while users in the Hetrec2011-movielens dataset are likely to view items from different angles.

5.5 Visualization (RQ4)

Lastly, we come to RQ4. In order to verify that MetaMF generates private item embeddings and RP models for users, we visualize the generated weights and item embeddings after reducing their dimension by t-SNE (van der Maaten and Hinton 2008) and normalizing them by mean and standard deviation,[6] where each point represents a user's weights or item embeddings. Because there are many items, we randomly select two items from each dataset for visualization. As shown in Fig. 2, MetaMF generates different weights and item embeddings for different users, which indicates that MetaMF has the ability to better capture the private factors for users. And we also notice the existence of many non-trivial clusters in each image, which shows that MetaMF is able to share information among users to take advantage of collaborative filtering. Compared to previous MF methods that share common item embeddings and RP models, MetaMF is very flexible.

[6]Here, $\mathrm{norm}(x) = \frac{x-\mu}{\sigma}$, where μ is the mean and σ is the standard deviation.

6 Conclusion

In this paper, we have studied matrix factorization methods for the rating prediction task. We have first argued that each user has her own views w.r.t. items and that a single common method/model is unlikely to satisfy all users. We have proposed a novel matrix factorization framework, named MetaMF. MetaMF first employs a collaborative memory module and a meta recommender module with a rise-dimensional generation strategy to generate private item embeddings and a rating prediction model for a user. Then MetaMF predicts the user's rating for a given item based on the generated item embeddings and rating prediction model. We conduct extensive experiments to validate the performance of MetaMF which can improve the performance of rating predictions by generating private item embeddings and rating prediction models.

The main limitation of MetaMF is that it requires users to have enough data for learning the private item embeddings and RP models. To generate item embeddings and RP models, the meta recommender module needs a large number of parameters and computations. As to our future work, we plan to enhance MetaMF for dealing with the user cold-start problem. We also would like to consider alternative CM and MR modules to reduce the number of parameters and further improve the performance. Additionally, we hope to incorporate side information and implicit feedback into MetaMF.

References

Bertinetto, L.; Henriques, J. F.; Valmadre, J.; Torr, P.; and Vedaldi, A. 2016. Learning feed-forward one-shot learners. In *NeurIPS*, 523–531.

Cantador, I.; Brusilovsky, P. L.; and Kuflik, T. 2011. Second workshop on information heterogeneity and fusion in recommender systems. In *HetRec2011*.

Chen, Y., and de Rijke, M. 2018. A collective variational autoencoder for top-n recommendation with side information. In *3rd Workshop on Deep Learning for Recommender Systems*. ACM.

Chen, F.; Dong, Z.; Li, Z.; and He, X. 2018a. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876*.

Chen, J.; Qiu, X.; Liu, P.; and Huang, X. 2018b. Meta multitask learning for sequence modeling. In *AAAI*.

Cheng, Z.; Ding, Y.; He, X.; Zhu, L.; Song, X.; and Kankanhalli, M. S. 2018a. Aˆ 3ncf: An adaptive aspect attention model for rating prediction. In *IJCAI*, 3748–3754.

Cheng, Z.; Ding, Y.; Zhu, L.; and Kankanhalli, M. 2018b. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *WWW*, 639–648.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. *JMLR* 9:249–256.

Goldberg, K.; Roeder, T.; Gupta, D.; and Perkins, C. 2001. Eigentaste: A constant time collaborative filtering algorithm. *Information retrieval* 4(2):133–151.

Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW*, 173–182.

He, X.; Du, X.; Wang, X.; Tian, F.; Tang, J.; and Chua, T. S. 2018. Outer product-based neural collaborative filtering. In *IJCAI*.

Hu, L.; Sun, A.; and Liu, Y. 2014. Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction. In *SIGIR*, 345–354.

Jia, X.; De Brabandere, B.; Tuytelaars, T.; and Gool, L. V. 2016. Dynamic filter networks. In *NeurIPS*, 667–675.

Kim, D.; Park, C.; Oh, J.; Lee, S.; and Yu, H. 2016. Convolutional matrix factorization for document context-aware recommendation. In *RecSys*, 233–240.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* 42(8):30–37.

Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, 426–434.

Krueger, D.; Huang, C.-W.; Islam, R.; Turner, R.; Lacoste, A.; and Courville, A. 2017. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*.

Lee, J.; Kim, S.; Lebanon, G.; Singer, Y.; and Bengio, S. 2016. Llorma: Local low-rank matrix approximation. *JMLR* 17(1):442–465.

Li, X., and She, J. 2017. Collaborative variational autoencoder for recommender systems. In *SIGKDD*, 305–314.

Li, P.; Wang, Z.; Ren, Z.; Bing, L.; and Lam, W. 2017. Neural rating regression with abstractive tips generation for recommendation. In *SIGIR*, 345–354.

Marlin, B. M. 2004. Modeling user rating profiles for collaborative filtering. In *NeurIPS*, 627–634.

Mnih, A., and Salakhutdinov, R. R. 2008. Probabilistic matrix factorization. In *NeurIPS*, 1257–1264.

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Ravi, S., and Larochelle, H. 2017. Optimization as a model for few-shot learning. In *ICLR*.

Salakhutdinov, R.; Mnih, A.; and Hinton, G. 2007. Restricted boltzmann machines for collaborative filtering. In *ICML*, 791–798.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2000. Application of dimensionality reduction in recommender system-a case study. In *WebKDD*.

Sedhain, S.; Menon, A. K.; Sanner, S.; and Xie, L. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*, 111–112.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *NeurIPS*, 4077–4087.

Strub, F.; Gaudel, R.; and Mary, J. 2016. Hybrid recommender system based on autoencoders. In *DLRS*, 11–16.

van der Maaten, L., and Hinton, G. 2008. Visualizing data using t-SNE. *JMLR* 9(Nov):2579–2605.

Vartak, M.; Thiagarajan, A.; Miranda, C.; Bratman, J.; and Larochelle, H. 2017. A meta-learning perspective on cold-start recommendations for items. In *NeurIPS*, 6904–6914.

Wang, C.; Liu, Q.; Wu, R.; Chen, E.; Liu, C.; Huang, X.; and Huang, Z. 2018. Confidence-aware matrix factorization for recommender systems. In *AAAI*.

Xiao, T.; Liang, S.; Shen, H.; and Meng, Z. 2019a. Neural variational hybrid collaborative filtering. In *WWW*.

Xiao, T.; Liang, S.; Shen, W.; and Meng, Z. 2019b. Bayesian deep collaborative matrix factorization. In *AAAI*.

Xu, Z.; van Hasselt, H. P.; and Silver, D. 2018. Meta-gradient reinforcement learning. In *NeurIPS*, 2396–2407.

Xue, H.-J.; Dai, X.; Zhang, J.; Huang, S.; and Chen, J. 2017. Deep matrix factorization models for recommender systems. In *IJCAI*, 3203–3209.

Yi, B.; Shen, X.; Liu, H.; Zhang, Z.; Zhang, W.; Liu, S.; and Xiong, N. 2019. Deep matrix factorization with implicit feedback embedding for recommendation system. *IEEE TII*.

Zhang, S.; Wang, W.; Ford, J.; and Makedon, F. 2006. Learning from incomplete ratings using non-negative matrix factorization. In *SDM*, 549–553.