# Mixture Matrix Approximation for Collaborative Filtering

Dongsheng Li, *Member, IEEE,* Chao Chen, Tun Lu, Stephen M. Chu, Ning Gu

**Abstract**—Matrix approximation (MA) methods are integral parts of today's recommender systems. In standard MA methods, only one feature vector is learned for each user/item, which may not be accurate enough to characterize the diverse interests of users/items. For instance, users could have different opinions on a given item, so that they may need different feature vectors for the item to represent their unique interests. To this end, this paper proposes a mixture matrix approximation (MMA) method, in which we assume that the user-item ratings follow mixture distributions and the user/item feature vectors vary among different stars to better characterize the diverse interests of users/items. Furthermore, we show that the proposed method can tackle both rating prediction and the top-N recommendation problems. Empirical studies on MovieLens, Netflix and Amazon datasets demonstrate that the proposed method can outperform state-of-the-art MA-based collaborative filtering methods in both rating prediction and top-N recommendation tasks.

**Index Terms**—Collaborative filtering, matrix approximation.

## 1 Introduction

Matrix approximation (MA) is one of the most successful collaborative filtering (CF) methods, in both rating prediction [3], [23], [26], [41], [49] and top-N recommendation [19], [34], [38], [47]. In collaborative filtering tasks, the user-item ratings (either multivariate or binary) are partially observed, and a general goal of MA methods is to recover the unobserved ratings based on the observed ratings. Generally, MA methods first learn a low-dimensional feature vector for every user/item from the observed ratings. Then, the unobserved ratings can be predicted by the dot products of the corresponding user feature vectors and item feature vectors [23]. User ratings on items are usually sparse, so that MA methods which can address the data sparsity issue by dimension reduction can achieve superior accuracy compared with other methods [39], [42], [44].

The user-item ratings are typically quantized in collaborative filtering tasks, e.g., 1-star to 5-star in the MovieLens 1M dataset, which causes the users' interests to be diverse on the same item [46]. For instance, given a particular item, the users who like the item will give 4/5-star ratings and the neutral users will give 3-star ratings, but meanwhile the users who dislike the item will give 1/2-star ratings. Therefore, a global matrix approximation model, i.e., using a globally optimized item feature vector for all kinds of users, cannot accurately represent the diverse interests of users on the item. Our case study on the MovieLens dataset shows that the optimal item feature vectors vary for different users and the globally optimized item feature vectors cannot achieve optimal recommendation accuracy for all users. Recently, categorical matrix approximation methods [5], [9] have been proposed to address the matrix approximation problem on quantized data. These methods can indeed improve the model performance over previous methods, but cannot address the issue mentioned above because they also learn global user/item feature vectors to recover unobserved ratings.

- D. Li is an adjunct professor with school of computer science, Fudan University and a research staff member with IBM Research - China, Shanghai, 201203, P. R. China. E-mail: dongshengli@fudan.edu.cn
- C. Chen and S. M. Chu are with IBM Research - China, Shanghai, 201203, P. R. China.E-mail: {cchao, schu}@cn.ibm.com
- T. Lu and N. Gu are with school of computer science, Fudan University, Shanghai, 201203, P. R. China.E-mail: {tunlu, ninggu}@fudan.edu.cn
- Tun Lu is the corresponding author.

This paper proposes a mixture matrix approximation (MMA) method, in which we assume that the user-item ratings follow mixture distributions over the stars and the user/item feature vectors vary among the stars to better characterize the diverse interests of users/items. More specifically, each user-item rating is described by a mixture of biased MA models, in which each biased MA model tries to learn the partial interests of users/items on a specific star. Then, for each rating, a mixture distribution is exploited to describe the relationship between the user/item/rating and different biased MA models. Therefore, we can have three variants of mixture matrix approximation method — the user-based MMA (U-MMA), the item-based MMA (I-MMA) and the rating-based MMA (R-MMA) by placing mixture assumptions over user ratings, item ratings, and individual ratings, respectively. In addition, we can naturally extend the proposed method from rating prediction to the top-N recommendation problem without much effort. To improve model performance and learning speed, we propose to pretrain the proposed MMA models, which can effectively solve the non-convex optimization problem associated with MMA. Based on the pretrained models, two learning methods are proposed to learn the probability to choose each component in the mixture model. Our empirical studies on the MovieLens, Netflix and Amazon datasets demonstrate that the proposed method can achieve higher accuracy compared with state-of-the-art MA-based collaborative filtering methods in both rating prediction (i.e., predict how a user will rate a movie or product) and top-N recommendation (i.e., predict if a user will rate a movie or buy a product or not) tasks while achieving high scalability.

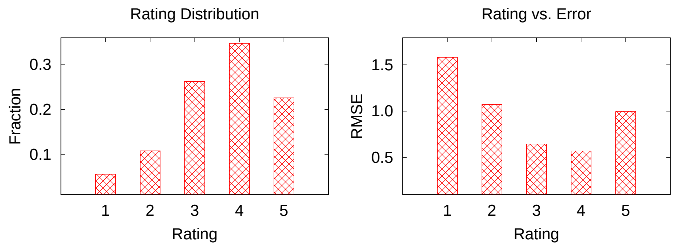

Fig. 1: The left figure shows the distribution of different ratings (1-star to 5-star) on the MovieLens 1M dataset. The right figure shows the rating specific RMSE, i.e., the RMSEs are computed based on all ratings of the same star.

## 2 Problem Formulation

This section first introduces the background of low-rank matrix approximation, and then motivates the targeted problem using three case studies on the MovieLens datasets.

### 2.1 Low-Rank Matrix Approximation

Given a user-item rating matrix $R \in \mathbb{R}^{m \times n}$, a general goal of matrix approximation is to find two rank $r$ feature matrix $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$ so that

$$R \approx \hat{R} = UV^T. \tag{1}$$

Here, we say $U$ is the user feature matrix and $V$ is the item feature matrix. Each row of $U$ or $V$ — $U_i$ or $V_j$ is the corresponding feature vector of the $i$-th user or the $j$-th item. Typically, $r$ is much lower than $m$ and $n$, so we can also call it low-rank matrix approximation.

To learn proper $U$ and $V$, we can minimize the discrepancy between the rating matrix $R$ and its approximation $\hat{R}$. The mean square error with $\ell_2$ regularization is one of the popular losses in collaborative filtering [23], [41], which can be described as follows:

$$L = \sum_{i,j} \mathbb{1}_{i,j}(R_{i,j} - \hat{R}_{i,j})^2 + \mu_1 ||U||_F^2 + \mu_2 ||V||_F^2. \tag{2}$$

Here, $R_{i,j}$ is the entry at the $i$-th row and $j$-th column of $R$. $\mathbb{1}_{i,j}$ is an indication function, which is 1 if $R_{i,j}$ is observed and 0 otherwise. $|| \cdot ||_F$ is the Frobenius norm, and $\mu_1$ and $\mu_2$ are the $\ell_2$ regularization coefficients of $U$ and $V$, respectively.

When the observed ratings are binary, i.e., implicit feedback of users on items, surrogate loss functions, e.g., exponential loss [28], are adopted to address the top-N recommendation problem. Therefore, we can change Equation 2 as follows:

$$L' = \sum_{i,j} \mathbb{1}_{i,j} \exp\{-R_{i,j}\hat{R}_{i,j}\} + \mu_1 ||U||_F^2 + \mu_2 ||V||_F^2. \tag{3}$$

### 2.2 Case Study

Here, we conduct three case studies on the MovieLens 1M dataset using the RSVD method [35]. In RSVD, we use 0.001 as the learning rate and 0.02 as the regularization coefficient. For each study, we randomly split the data into a training set and a test set by the ratio of 9:1, and stop training when the test errors start to grow.

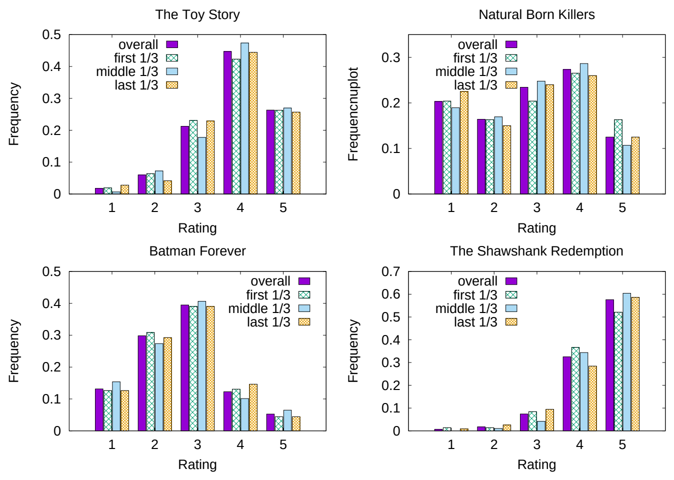

Fig. 2: The rating distributions of four movies on the MovieLens 1M dataset. "Overall" means the distribution over all ratings, "first 1/3" means the distribution of the earliest 1/3 ratings for the movie, and so forth. The x-axis is the stars of the ratings and the y-axis is the frequency of the ratings for each movie.

#### 2.2.1 *Case Study 1: Rating vs. Error*

This case study analyzes the relationship between rating distribution and recommendation accuracy on the MovieLens 1M dataset. As shown in Figure 1 (left), the ratings are actually highly biased, i.e., there are much more 3/4/5-star ratings than 1/2-star ratings. In addition, we can see from Figure 1 (right) that the recommendation accuracy for 4-star ratings (the most frequent ratings) is much higher than the accuracy for 1-star ratings (the most infrequent ratings). This study indicates that the globally learned MA model is biased towards frequent ratings and thus achieves suboptimal performance for other ratings, i.e., user interests on 1/2/3/5-star movies are suboptimal. Therefore, it is necessary to learn different biased MA models towards different stars to better characterize the diverse interests of users.

#### 2.2.2 *Case Study 2: The Mixture Distribution*

This case study analyzes the rating distribution of individual movies in the MovieLens 1M dataset. Here, we select four movies from the dataset and compare their ratings distributions in different time intervals. More specifically, we divide the ratings of each movie into three part by chronological order, i.e., the earliest 1/3 ratings, the middle 1/3 ratings and the latest 1/3 ratings. As shown in Figure 2, the rating distributions of overall ratings and partial ratings are very similar for all four movies, i.e., the probability of a particular rating occurring can be regarded as constant. Therefore, we can conclude that 1) the mixture distribution of each movie's ratings is very stable over time so that we can learn a mixture model and use it for future recommendations and 2) the rating distributions of the movies are very different from each other, which indicates that different mixture model parameters should be used to describe different users/items.

#### 2.2.3 *Case Study 3: Rating vs. Model*

In this study, we first train an optimal MA model using RSVD with lowest RMSE on the test set. Then, we fix the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2955100, IEEE Transactions on Knowledge and Data Engineering

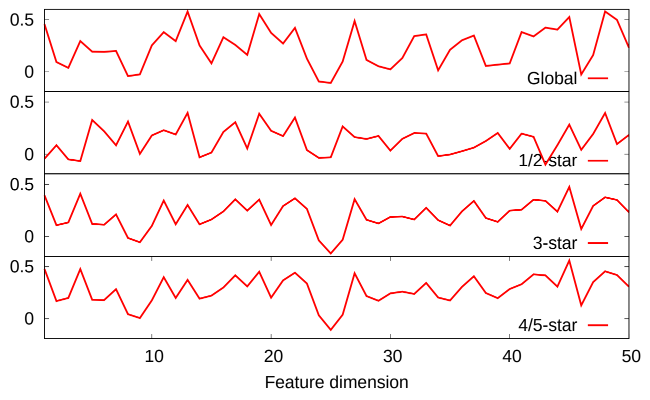

Fig. 3: Case study: comparison of parameter values for the movie "Toy Story" optimizing towards different stars. "Global" means the optimal parameter values based on all ratings of the movie, "1/2-star" means the optimal parameter values based on 1/2-star ratings only, and so forth.

user feature vectors and update the item feature vector for the movie "Toy Story" based on specific stars. For instance, "1/2-star" means we only train the optimal item feature vector based on 1/2-star ratings of "Toy Story".

Figure 3 shows the learned item feature vectors for different cases with rank = 50. We can see from the results that the optimal feature vectors are indeed different among the stars. For instance, in the first dimension, the parameter value for 1/2-star is negative, but the values are positive for 3-star and 4/5-star. Besides, the global model shows larger variance among the parameter values, which should be due to a compromise among diverse user interests.

In addition, we also compared the accuracy of the global model and the biased models learned on specific stars, and we observe a huge performance improvement of the biased models on the specific stars. For instance, the global model achieves a RMSE of 0.7999 on 4/5-star, but the RMSE on 4/5-star is 0.6326 if we train the model using only 4/5-star ratings. This further confirms that globally optimized matrix approximation models cannot capture the diverse characteristics of users/items and thus achieve inferior performances in recommendation tasks.

## 3 Mixture Matrix Approximation

Following the PMF method [41], we describe the user-item ratings using a probabilistic model with Gaussian noise. As shown in Figure 4, the conditional distribution over the observed user-item ratings for the mixture matrix approximation can be defined as follows:

$$\Pr(R|U,V,\Pi,\sigma^2) = \prod_{i=1}^{m}\prod_{j=1}^{n}[\sum_{k\in\mathbb{S}} \Pi_{i,j}^k \mathcal{N}(R_{i,j}|U_i^k(V_j^k)^T,\sigma^2)]^{\mathbb{1}_{i,j}}. \tag{4}$$

Here, $\mathcal{N}(\cdot|\mu,\sigma^2)$ denotes the probability density function of a Gaussian distribution with mean $\mu$ and variance $\sigma^2$. $\mathbb{S}$ denotes the set of possible ratings, e.g., 1-star to 5-star in rating prediction and 0/1 in top-N recommendation. For simplicity, the term "rating" means both 1-5 star ratings and binary ratings unless otherwise specified. $\Pr(R_{i,j}=k)=\Pi_{i,j}^k$ for any $k\in\mathbb{S}$, i.e., $\Pi_{i,j}^k$ denotes the probability of choosing the model focused on $k$-star ratings. $U^k$ and $V^k$ are the feature matrices of the matrix approximation model focusing on $k$-star ratings, and $U_i^k$ and $V_j^k$ denote the feature vectors for the $i$-th user and $j$-th item, respectively. $\mathbb{1}_{i,j}$ is an indication function, which is 1 if $R_{i,j}$ is observed and 0 otherwise. Intuitively, the MMA method tries to learn $|\mathbb{S}|=K$ different biased MA models, each of which can be very accurate on a specific star. Then, the final prediction is a weighted sum of the predictions from different biased MA models.

We adopt a zero mean isotropic Gaussian prior [14], [41] on the user and item feature vectors as follows:

$$\begin{aligned}\Pr(U^k|\sigma_U^2) &= \prod_{i=1}^{m}\mathcal{N}(U_i^k|0,\sigma_U^2 I),\\ \Pr(V^k|\sigma_V^2) &= \prod_{j=1}^{n}\mathcal{N}(V_j^k|0,\sigma_V^2 I).\end{aligned}$$

We place a multinomial prior on $\Pi$, so the probability of $\Pi$ can be given as follows:

$$\Pr(\Pi)=\prod_{i=1}^{m}\prod_{j=1}^{n}(\Pi_{i,j})^{\mathbb{1}_{i,j}}.$$

Here, $\Pi_{i,j}$ denotes the multinomial distribution corresponding to user $i$ and item $j$. $\Pi_{i,j}$ should change when different variants of MMA are adopted, e.g., $\Pi_{i,j}$ does not change with $j$ if we use U-MMA.

Assuming that $U$, $V$ and $\Pi$ are independent to each other, then the posterior distribution over the user and item feature matrices $U, V$ and rating distribution $\Pi$ can be given as follows:

$$\begin{aligned}&\Pr(U,V,\Pi|R,\sigma^2,\sigma_U^2,\sigma_V^2)\propto\\ &\Pr(R|U,V,\Pi,\sigma^2)\Pr(U|\sigma_U^2)\Pr(V|\sigma_V^2)\Pr(\Pi).\end{aligned} \tag{5}$$

Then, assuming that $\sigma^2$, $\sigma_U^2$ and $\sigma_V^2$ are constants, we can derive the log of the posterior distribution over $U$, $V$ and $\Pi$ as follows:

$$\begin{aligned}l &= \ln\Pr(U,V,\Pi|R,\sigma^2,\sigma_U^2,\sigma_V^2)\\ &\propto \sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}[\ln\sum_{k\in\mathbb{S}}\Pi_{i,j}^k\mathcal{N}(R_{i,j}|U_i^k(V_j^k)^T,\sigma^2)] -\\ &\frac{1}{2\sigma_U^2}\sum_{k\in\mathbb{S}}\sum_{i=1}^{m}(U_i^k)^2-\frac{1}{2\sigma_V^2}\sum_{k\in\mathbb{S}}\sum_{j=1}^{n}(V_i^k)^2+\\ &\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}\ln\Pi_{i,j}+C.\end{aligned} \tag{6}$$

Here, $C$ is a constant that does not depend on any parameters, which will be omitted in the objective. There are summations within the logarithm operations in Equation 6, which makes the above optimization problem be difficult to solve. Therefore, we first obtain the lower bound of Equation 6 using Jensen's inequality, and then we minimize the negative log likelihood as follows:

$$\begin{aligned}l' = &\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}[\sum_{k\in\mathbb{S}}\Pi_{i,j}^k(R_{i,j}-U_i^k(V_j^k)^T)^2]+\frac{\sigma^2}{\sigma_U^2}\sum_{k\in\mathbb{S}}\sum_{i=1}^{m}(U_i^k)^2\\ &+\frac{\sigma^2}{\sigma_V^2}\sum_{k\in\mathbb{S}}\sum_{j=1}^{n}(V_i^k)^2-2\sigma^2\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}\ln\Pi_{i,j}.\end{aligned}$$

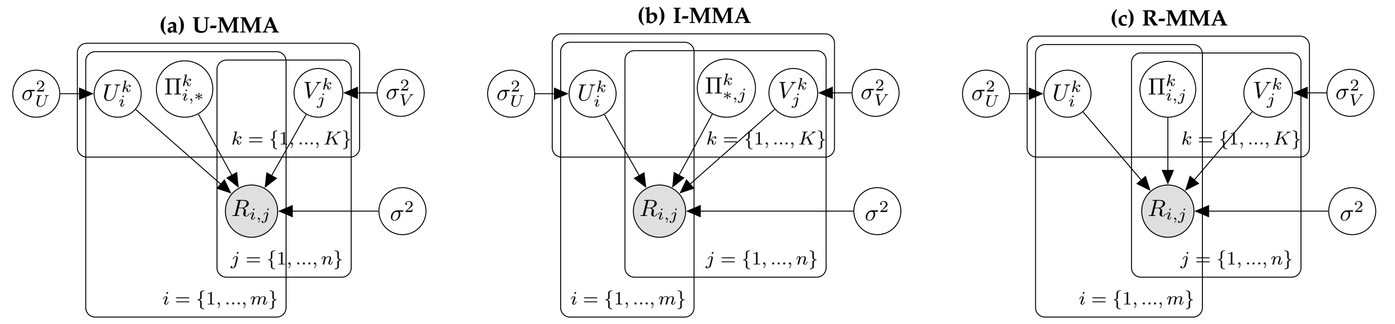

Fig. 4: The graphical model for the proposed mixture matrix approximation method. Here, we show the plate notations of three variants of MMA, i.e., the user-based MMA (U-MMA), the item-based MMA (I-MMA) and the rating-based MMA (R-MMA) by placing mixture assumptions over user ratings, item ratings, and individual ratings, respectively.

Then, by considering $\sigma^2/\sigma_U^2$, $\sigma^2/\sigma_V^2$ and $2\sigma^2$ as the regularization coefficients for $U$, $V$ and $\Pi$, we can obtain the final optimization objective for the proposed mixture matrix approximation method as follows:

$$L = \sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}\big[\sum_{k\in\mathbb{S}}\Pi_{i,j}^k(R_{i,j} - U_i^k(V_j^k)^T)^2\big] + \mu_1\sum_{k\in\mathbb{S}}\sum_{i=1}^{m}(U_i^k)^2 + \mu_2\sum_{k\in\mathbb{S}}\sum_{j=1}^{n}(V_i^k)^2 - \mu_3\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}\ln\Pi_{i,j}. \tag{7}$$

Note that, different variants of the mixture matrix approximation method can be derived by placing the mixture model assumption $\Pi$ over different aspects, i.e., assuming that the ratings of each user follow a mixture model, the ratings of each item follow a mixture model, or each rating follows a mixture model. Thus, we can opt for different variants of MMA by using different mixture models (Figure 4).

The above loss function (Equation 7) can apply to general CF tasks, e.g., both rating prediction and top-N recommendation. However, it may not be optimal due to the implicit feedback issue in top-N recommendation task [19], [34], [38]. Therefore, we introduce surrogate loss functions, e.g., Exponential loss [28], into Equation 7 and formulate the loss function for top-N recommendation as follows:

$$L' = \sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}\big[\sum_{k\in\mathbb{S}}\Pi_{i,j}^k\exp\{-R_{i,j}(U_i^k(V_j^k)^T)\}\big] + \mu_1\sum_{k\in\mathbb{S}}\sum_{i=1}^{m}(U_i^k)^2 + \mu_2\sum_{k\in\mathbb{S}}\sum_{j=1}^{n}(V_i^k)^2 - \mu_3\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbb{1}_{i,j}\ln\Pi_{i,j}.$$

Note that other surrogate loss functions, e.g., mean square loss and log loss, are also popular in ranking tasks [25], [28] and could be applied in the above loss function for top-N recommendation. Therefore, the optimal surrogate loss should be determined empirically.

## 4 Learning MMA Models

Stochastic gradient descent (SGD) is one of the possible methods to minimize the non-convex objective defined in Equation 7. However, our empirical studies show that directly minimizing Equation 7 using SGD will easily be trapped by local minimum and achieve very poor generalization performance (Figure 5). Therefore, we propose to pretrain each component in MMA. More specifically, we proposed a weighted matrix approximation method to help pretrain the biased MA models, in which we set larger weights for $k$-star ratings if we want the learned MA model to be biased towards $k$-star ratings. Then, two learning methods are proposed to learn the probability of choosing each component in the mixture model.

### 4.1 The Weighted Matrix Approximation

Given a targeted star $k \in \mathbb{S}$, let $\hat{R}^k$ be the biased MA model towards $k$-star ratings. The weighted matrix approximation should be performed by solving the following problem:

$$\hat{R}^k = \arg\min_{R'} ||W^k \otimes (R - R')||_F^2. \tag{8}$$

Here, $W^k$ is the weight matrix for star $k$, and $\otimes$ denotes the element-wise product.

To learn biased MA models, the proposed weighted matrix approximation method should satisfy the following two requirements: 1) ratings that are close to $k$-star in the training set should be given larger weights than far away ratings; 2) users/items who have more $k$-star ratings should be given larger weights than the other users/items. Considering the above two requirements, we propose the weighting strategy on training example $R_{i,j}$ for $\hat{R}^k$ as follows:

$$W_{i,j}^k = (1 - \alpha|R_{i,j} - k|)F(k) + \beta. \tag{9}$$

$\alpha, \beta > 0$ are two hyperparameters for the proposed weighted MA method. $F(k)$ stands for the frequency of $k$-star ratings for the user and/or the item, e.g., the frequency of $k$-star ratings from user $i$ if we use user-based MMA. Here, $(1 - \alpha|R_{i,j} - k|)$ ensures that ratings that are closer to $k$-star are given larger weights, and $F(k)$ ensures that users/items who have more $k$-star ratings are given larger weights. $\beta$ is a constant to prevent $W$ from being too small, which is set to 0.1 empirically in this paper. Note that other distance measures between $R_{i,j}$ and $k$ could be adopted here instead of the absolute function, e.g., Gaussian kernel and Laplacian kernel are both popular when little prior knowledge is known about the data [18]. However, our studies show that absolute distance is more desirable in collaborative filtering due to better empirical performances (Figure 10).

After defining $W^k_{i,j}$, we can learn the biased MA model for rating $k$ — $\hat{R}^k$ using SGD as follows:

$$U^k_i \leftarrow U^k_i - \lambda W^k_{i,j}((\hat{R}_{i,j} - R_{i,j})V^k_j + \mu_1 U^k_i),$$
$$V^k_j \leftarrow V^k_j - \lambda W^k_{i,j}((\hat{R}_{i,j} - R_{i,j})U^k_i + \mu_2 V^k_j).$$

### 4.2 Learning the Mixture Distributions

Since we have different variants of MMA, we propose two different learning methods: 1) SGD-based method to learn the models for U-MMA and I-MMA efficiently and 2) classification-based method to learn the models for R-MMA, in which we learn classification models to predict the probability of choosing each component in the mixture model. The reason for the classification-based method is that the probability distributions are rating-specific for R-MMA, so that we cannot have the learned distributions for test ratings in R-MMA if we use SGD because test ratings do not exist in training data.

#### *4.2.1 SGD-based Method*

After learning the biased MA models for each star, we can set the initial values of all the parameters in $U$ and $V$. Meanwhile, $\Pi$ can be initialized based on the empirical rating distributions of users/items, e.g., $\Pi^k_{i,j} = F(k)$. Then, we can learn $\Pi$ and update $U$ and $V$ using standard SGD method. For each training example $R_{i,j}$ and each star $k$, the gradient update rules for $U^k$ and $V^k$ can be described as follows:

$$U^k_i \leftarrow U^k_i - \lambda_1((\hat{R}^k_{i,j} - R_{i,j})V^k_j \Pi^k_{i,j} + \mu_1 U^k_i).$$
$$V^k_j \leftarrow V^k_j - \lambda_1((\hat{R}^k_{i,j} - R_{i,j})U^k_i \Pi^k_{i,j} + \mu_2 V^k_j).$$

The gradient update rules for $\Pi^k$ in U-MMA and I-MMA can be described as follows:

$$\Pi^k_{i,*} \leftarrow \Pi^k_{i,*} - \lambda_2((R_{i,j} - \hat{R}^{(k)}_{i,j})^2 - \mu_3/\Pi^k_{i,*}).$$
$$\Pi^k_{*,j} \leftarrow \Pi^k_{*,j} - \lambda_2((R_{i,j} - \hat{R}^{(k)}_{i,j})^2 - \mu_3/\Pi^k_{*,j}).$$

#### *4.2.2 Classification-based Method*

We can use classification algorithms to learn $\Pi$ by regarding the probabilities in the mixture distributions as classification probabilities. Many off-the-shelf multi-class classification methods, e.g., random forest (RF) [8], multinomial logistic regression (MLR) [7] and deep neural network (DNN) [12], can be applied. Here, we choose DNN as an example to illustrate how to apply classification methods to learn $\Pi$ in MMA.

Deep neural network (DNN) with rectified linear unit (ReLU) and dropout has been successfully applied in speech recognition [12], which is efficient due to the adoption of ReLU [15] and generalizes well due to the adoption of dropout [43]. In this paper, we can use DNN as the classifier to determine the probabilities of choosing different biased MA models given their scores. More specifically, the input of the DNN is the recommendation scores from all biased MA models and the output of the DNN is the probabilities of choosing different biased MA models.

The output of the $t$-th layer of the DNN can be described as follows [12]:

$$\vec{y}_t = f(\frac{1}{1-d}\vec{y}_{t-1} \otimes \vec{m} \cdot \vec{x} + \vec{b}), \tag{10}$$

where $d$ is the dropout probability, $\vec{m} \sim B(1, 1-d)$ is the binary mask indicating which units are not dropped out, $\vec{x}$ represents the weights and $\vec{b}$ represents the bias in this layer. Here, "$\otimes$" denotes element-wise product and "$\cdot$" denotes dot product. In particular, the activation function $f$ is defined as follows: $f(x) = max\{0, x\}$. For simplicity, we use the same number of units for all hidden layers. We use the $\ell_2$ norm as the regularization term and Adam [21] to adaptively tune the learning rate in SGD. The output layer of the DNN is transformed using the softmax function to obtain the probability of choosing the biased MA model towards $k$-star as follows:

$$\Pi^k_{i,j} = \frac{\exp\{y(k)\}}{\sum_{a=1}^{K}\exp\{y(a)\}}. \tag{11}$$

Here, $y(k)$ is the output of the DNN for the $k$-th class. $\Pi^k_{i,j}$ is the probability of choosing the biased MA model towards $k$-star. Note that $U$ and $V$ need not be updated in the classification-based method.

### 4.3 Model Prediction

After learning $U$, $V$ and $\Pi$, we can predict user ratings on unseen items using the mixture model. More specifically, we can compute the recommendation scores as follows:

$$\hat{R}_{i,j} = \sum_{k\in\mathbb{S}} \Pi^k_{i,j} U^k_i (V^k_j)^T. \tag{12}$$

$\Pi^k_{i,j}$ denotes the probability of choosing the biased MA model towards $k$-star in R-MMA, so that we should replace it with $\Pi^k_{i,*}$/ $\Pi^k_{*,j}$ if we use U-MMA/I-MMA, respectively.

### 4.4 Efficiency Analysis

Compared with stand alone methods, e.g., RSVD [35], the proposed method requires higher overall computation overhead due to the learning of multiple biased MA models and mixture distributions. However, the extra overhead of MMA is linear compared with RSVD, e.g., around K times computations on a dataset with $K$ stars. In addition, the above two steps can both be performed in parallel to improve the scalability.

For the weighted MA method, the biased MA models can be easily learned in parallel, because they have no dependencies on each other. For each biased MA model, we can further increase the scalability by adopting other learning algorithms, e.g., alternating least squares [19], which are amenable to parallelization.

For learning the mixture model distributions, the SGD-based method cannot be easily parallelized. However, since all the biased MA models and the mixture distributions are initialized properly, the learning process will converge very fast, e.g., within 50 epochs in our empirical studies. For the classification-based method, the scalability depends on the selected classification method itself. For instance, in the learning of DNN, the parameter matrices can be updated in parallel by adopting parallel matrix computations.

Note that the proposed method can be applied in online recommendation for relatively stable users and items because each recommendation score can be computed via a few dot products, which is similar to existing matrix approximation methods [27], [29], [35]. For fast changing users and items, we can adopt existing online matrix approximation methods [17] to update the biased MA models in MMA. Then, we can choose R-MMA to combine the biased MA models because the classification model can be re-trained online via cost-sensitive online classification methods [45].

## 5 Generalization Error Analysis

This section analyzes the generalization error bounds of the weighted matrix approximation method and the proposed MMA method.

### 5.1 The Weighted Matrix Approximation Method

Since many ratings have equal weights in the proposed weighted MA method, we can ignore the regularization term and rephrase the loss function as follows:

$$\mathcal{L}'(R, \hat{R}^k) = \sum_{s=1}^{K'} \frac{\lambda_s}{|\Omega_s|} \sum_{(i,j)\in\Omega_s} (R_{i,j} - \hat{R}^k_{i,j})^2, \qquad (13)$$

where $\Omega_s$ is the set of ratings with equal weight. Without loss of generality, we can assume that $\lambda_1 + ... + \lambda_{K'} = 1$, because we can scale Equation 13 with a scaler if $\lambda_1 + ... + \lambda_{K'} \neq 1$. Here, we prove that the weighted loss function can achieve lower generalization error bound, i.e., potentially better generalization performance, than unweighted loss function in the following Theorem 1. The following results are adopted in the proof of Theorem 1.

**Lemma 1** (Markov's Inequality)**.** *Let $X$ be a real-valued non-negative random variable. Then, for any $\epsilon > 0$, $\Pr(X \geq \epsilon) \leq \frac{E[X]}{\epsilon}$.*

**Lemma 2** (Hoeffding's Lemma)**.** *Let $X$ be a real-valued random variable with zero mean and $\Pr(X \in [a, b]) = 1$. Then, for any $z \in \mathbb{R}$, $\mathrm{E}\left[e^{zX}\right] \leq \exp\left(\frac{1}{8}z^2(b-a)^2\right)$.*

**Theorem 1.** *Let $\mathcal{L} = 1/|\Omega| \sum_{(i,j)\in\Omega}(R_{i,j} - \hat{R}^s_{i,j})^2$, $\mathcal{L}'$ be defined as in Equation 13, and $E[\mathcal{L}] = E[\mathcal{L}'] = \mu$. For any $\epsilon > 0$, if $\Pr[|\mathcal{L}' - \mu| < \epsilon] \geq 1 - \delta'$ and $\Pr[|\mathcal{L} - \mu| < \epsilon] \geq 1 - \delta$, then $\delta' \leq \delta$.*

*Proof.* Based on Markov's inequality and Hoeffding's Lemma, for any $\epsilon, t > 0$, we have

$$\Pr[\mathcal{L} - \mu \geq \epsilon] = \Pr[e^{t(\mathcal{L}-\mu)} \geq e^{t\epsilon}] \overset{(a)}{\leq} \frac{E[e^{t(\mathcal{L}-\mu)}]}{e^{t\epsilon}} \overset{(b)}{\leq} \frac{e^{\frac{1}{8}t^2(a-b)^2}}{e^{t\epsilon}},$$

where $\mu$ is the expectation of $\mathcal{L}$, $a = \sup\{\mathcal{L} - \mu\}$ and $b = \inf\{\mathcal{L} - \mu\}$. (a) holds due to Markov's inequality, and (b) holds due to Hoeffding's Lemma. Similarly, we have

$$\Pr[\mu - \mathcal{L} \geq \epsilon] \leq e^{\frac{1}{8}t^2(a-b)^2}/e^{t\epsilon}. \qquad (14)$$

Combining the above two inequalities, we have

$$\Pr[|\mathcal{L} - \mu| < \epsilon] \geq 1 - 2e^{18t^2(a-b)^2}/e^{t\epsilon},$$

i.e.,

$$\delta = 2e^{\frac{1}{8}t^2(a-b)^2}/e^{t\epsilon}. \qquad (15)$$

Let $\mathcal{L}'_s = \frac{1}{|\Omega_s|}\sum_{(i,j)\in\Omega_s}(R_{i,j} - \hat{R}^{(s)}_{i,j})^2$ $(s \in \{1, ..., K'\})$. Then, we know that $\mathcal{L}'_s$ and $\mathcal{L}$ have the same expectation $\mu$. Therefore, we can derive $\delta'$ for $\mathcal{L}'$ as follows:

$$\begin{aligned}&\Pr[\sum_{s\in\{1,...,K'\}} \lambda_s \mathcal{L}'_s - \mu \geq \epsilon] \\ &\overset{(a)}{\leq} E[e^{t(\sum_s \lambda_s \mathcal{L}'_s - \mu)}]/e^{t\epsilon} = E[e^{t(\sum_s \lambda_s(\mathcal{L}'_s - \mu))}]/e^{t\epsilon} \\ &\overset{(b)}{\leq} \sum_s \lambda_s E[e^{t(\mathcal{L}'_s - \mu)}]/e^{t\epsilon} \overset{(c)}{\leq} \sum_s \lambda_s e^{\frac{1}{8}t^2(a_s - b_s)^2}/e^{t\epsilon},\end{aligned}$$

where (a) holds due to Markov's inequality, (b) holds due to the convexity of exponential function, and (c) holds due to Hoeffding's Lemma. $a_s = \sup\{\mathcal{L}'_s - \mu\}$ and $b_s = \inf\{\mathcal{L}'_s - \mu\}$. Then, we have

$$\delta' = 2\sum_{s\in\{1,...,K'\}} \lambda_s \frac{e^{\frac{1}{8}t^2(a_s-b_s)^2}}{e^{t\epsilon}}. \qquad (16)$$

$\forall s \in \{1, ..., K'\}$, since $\Omega_s \subseteq \Omega$, we know $\sup\{\mathcal{L}'_s\} \leq \sup\{\mathcal{L}\}$ and $\inf\{\mathcal{L}'_s\} \geq \inf\{\mathcal{L}\}$. Therefore, $\sup\{\mathcal{L}'_s - \mu\} \leq \sup\{\mathcal{L} - \mu\}$ and $\inf\{\mathcal{L}'_s - \mu\} \geq \inf\{\mathcal{L} - \mu\}$, i.e., $a_s \leq a$ and $b_s \geq b$. Then, we have $(a - b)^2 \geq (a_s - b_s)^2$, i.e., $\frac{e^{\frac{1}{8}t^2(a_s-b_s)^2}}{e^{t\epsilon}} \leq \frac{e^{\frac{1}{8}t^2(a-b)^2}}{e^{t\epsilon}}$ $\forall s \in \{1, ..., K'\}$. Then, by the assumption that $\sum_s \lambda_s = 1$, we can conclude that $\delta' \leq \delta$. □

**Remark.** The above theorem demonstrates that $\mathcal{L}'$ will be close to its expectation $\mu$ with higher probability than $\mathcal{L}$, i.e., minimizing $\mathcal{L}'$ can achieve small generalization error with higher probability than minimizing $\mathcal{L}$. In other words, the proposed weighted loss function (Equation 13) can be better estimation of test error than unweighted loss function.

Note that the above theorem can be applied to general weighted matrix approximation methods. However, different weighting strategies can derive different $\delta'$, i.e., a well-designed weighting strategy can achieve better generalization performance than random weighting because a well-designed weighting strategy can achieve smaller $\delta'$.

### 5.2 The MMA Method

Similar to the weighted MA method, we can rephrase the loss function of the proposed MMA method as a weighted combination of loss functions as follows:

$$\begin{aligned}\mathcal{L}''(R, \hat{R}) &= \frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}\sum_k W^k_{i,j}(R_{i,j} - \hat{R}^k_{i,j})^2 \\ &\propto \sum_{x=1}^{K''} \frac{\lambda_x}{|\Omega_x|}\sum_{(i,j)\in\Omega_x}(R_{i,j} - \hat{R}^{s_x}_{i,j})^2. \qquad (17)\end{aligned}$$

Here, $\Omega_x$ can be regarded as the set of entries sharing the same weight in Equation 12. $s_x$ stands for the star that the model biased towards for the entries in $\Omega_x$. Again, we can assume that $\lambda_1 + ... + \lambda_{K''} = 1$ here.

Similarly, we can prove that $\mathcal{L}''$ can achieve lower generalization error bound, i.e., potentially better generalization performance, than $\mathcal{L}$ in the following Theorem.

**Theorem 2.** *Let $\mathcal{L} = 1/|\Omega| \sum_{(i,j)\in\Omega}(R_{i,j} - \hat{R}_{i,j})^2$, $\mathcal{L}''$ be defined as in Equation 17, and $E[\mathcal{L}] = E[\mathcal{L}''] = \mu$. For any $\epsilon > 0$, if $\Pr[|\mathcal{L}'' - \mu| < \epsilon] \geq 1 - \delta''$ and $\Pr[|\mathcal{L} - \mu| < \epsilon] \geq 1 - \delta$, then $\delta'' \leq \delta$.*

*Proof.* Proof can be similarly derived as Theorem 1. □

## 6 EXPERIMENTS

This section first describes our experimental setup. Then, we present the empirical results of the proposed mixture matrix approximation method on well-known real-world datasets, and compare its performance with state-of-the-art methods.

### 6.1 Experimental Setup

#### 6.1.1 Dataset Description.

We evaluate the proposed MMA method using five well-known real-world datasets: 1) MovieLens 100K dataset ($\sim$100,000 ratings from 943 users on 1682 movies); 2) MovieLens 1M dataset ($\sim$1 million ratings from 6,040 users on 3,706 movies); 3) MovieLens 10M dataset ($\sim$10 million ratings from 69,878 users on 10,677 movies); 4) Netflix Prize dataset ($\sim$100 million ratings from 480,189 users on 17,770 movies); 5) Amazon Instant Video Dataset (37,126 ratings from 5,130 users on 1,685 products). For rating prediction, we predict how will a user rate a movie/product. For top-N recommendation, we predict if a user will rate a movie/buy a product or not [20], [28]. For each experiment, we randomly split the dataset into a training set and a test set by the ratio of 9:1 unless otherwise specified. All the results are reported by averaging the numbers over five different random splits unless otherwise specified.

#### 6.1.2 Hyperparameter Setting.

For learning the biased matrix approximation models, we use the learning rate $\lambda$ = 1e-3 and regularization coefficients $\mu_1 = \mu_2 = 0.02$. We set $\alpha = 0.1$ and $\beta = 0.2$ in Equation 9 unless otherwise specified. Convergence is reached if the training RMSE reduction is less than 2e-4 or the number of iterations reaches 500. For learning the mixture model distributions using SGD, we use the learning rates $\lambda_1$ = 1e-5 and $\lambda_2$ = 1e-6 and $\mu_1 = \mu_2 = 0.02$ and $\mu_3 = 0.001$. Convergence is reached if the training RMSE reduction is less than 1e-6 or the number of iterations reaches 100. The hyperparameters of the classification-based method are chosen based on the sensitivity analysis on the training sets from MovieLens 1M dataset in Section 6.4. The hyperparameters of the compared method are adopted from the original papers because all the compared methods evaluated their methods on the same datasets. The hyperparameters of the proposed method are fixed for all the comparisons with state-of-the-art methods, i.e., better results could be obtained if we tune hyperparameters for each dataset. Note that although the proposed method has more hyperparameters than standalone methods, e.g., SMA [29], but the hyperparameters tuned on a small dataset, e.g., MovieLens 1M, can achieve better performance than state-of-the-art methods as demonstrated in Table 3 and Figure 14. Therefore, the proposed method can be efficiently applied on large datasets, in which the hyperparameter tuning can be performed on subsets of the data.

#### 6.1.3 Compared Methods.

We compare the proposed method with two kinds of state-of-the-art methods: a) categorical matrix approximation methods and b) state-of-the-art matrix approximation-based collaborative filtering methods.

The proposed MMA method is compared with the following two categorical matrix approximation methods. 1) Categorical matrix completion (CMC) [9] maximizes the likelihood ratio with nuclear norm constraint and maps the entries through multinomial logistic regression link functions. 2) Matrix completion from quantized measurements (MCQM) [5] considers maximum likelihood estimation under a constraint on the entry-wise infinity-norm and an exact rank constraint. There are different variants of MCQM, and we compare MMA with "Alg. 3 (log-barrier + unknown bins): logistic" which achieved the best empirical performance in the original paper.

For rating prediction, we compare the proposed method with the following six state-of-the-art matrix approximation-based methods. 1) DFC [32] uses a divide and conquer method on matrix factorization to improve both scalability and accuracy; 2) GSMF [48] applies a group sparsity regularization when learning user/item features in matrix approximation. 3) LLORMA [26] first learns biased models by weighted matrix approximation and then combines the approximations from different biased models using kernel smoothing. 4) WEMAREC [11] combines different biased matrix approximation models from co-clustering-based approximation using weighted average to achieve higher accuracy; 5) SMA [29] derives a stable matrix approximation problem that can achieve good generalization performance and recommendation accuracy. 6) MRMA [27] combines a set of low-rank matrix approximation with different rank values to achieve better accuracy.

For top-N recommendation, we compare the proposed method with the following five matrix approximation-based methods. 1) WRMF [19] sets higher weights to positive ratings and lower weights for unknown ratings in matrix factorization to address the implicit feedback issue; 2) BPR [38] defines a pair-wise loss function for top-N recommendation problem. Different versions of BPR methods were proposed in their paper, and we compare MMA with the BPR-MF; 3) SLIM [33] generates top-N recommendations by aggregating weighted user ratings learned by solving an $L_1$ and $L_2$ regularized optimization problem. 4) AOBPR [37] improves the classic BPR method by oversampling informative pairs to achieve faster convergence and higher accuracy; 5) eALS [17] adopts a non-uniform weighting on missing ratings and proposes an element-wise Alternating Least Squares (eALS) technique to improve both effectiveness and efficiency of matrix factorization.

#### 6.1.4 Evaluation Metrics.

For rating prediction, we evaluate the proposed method using the following two popular metrics. 1) Root mean square error (RMSE), which is defined as follows: RMSE($\hat{R}$) = $\sqrt{1/|\Omega| \sum_{(i,j)\in\Omega} (R_{i,j} - \hat{R}_{i,j})^2}$, in which $\Omega$ stands for the set of examples in the test set. 2) Mean average error (MAE), which is defined as follows: MAE($\hat{R}$) = $1/|\Omega| \sum_{(i,j)\in\Omega} |R_{i,j} - \hat{R}_{i,j}|$.

For top-N recommendation, we evaluate the proposed MMA method using NDCG@N and Precision@N. 1) NDCG@N = DCG@N/IDCG@N, in which DCG@N =$\sum_{i=1}^{n} (2^{rel_i} - 1)/log_2(i+1)$ and IDCG@N is the value of DCG@N with optimal ranking ($rel_i = 1$ if the user rated the $i$-th recommended item and $rel_i = 0$ otherwise); 2) Precision@N = $|I_r \cap I_u|/|I_r|$, in which $I_r$ is the list of top-N recommendations and $I_u$ is the list of items that $u$ has rated.

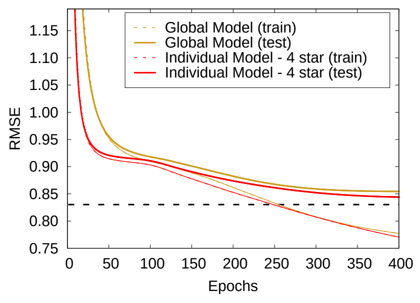

Fig. 5: Training and test errors vs. epochs of the globally trained mixture model and the biased MA model towards 4-star trained individually on the MovieLens 1M dataset.

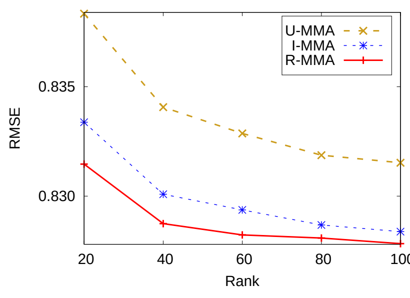

Fig. 6: Test RMSE comparison among U-MMA, I-MMA and R-MMA with ranks ranging from 20 to 100 on the MovieLens 1M dataset.

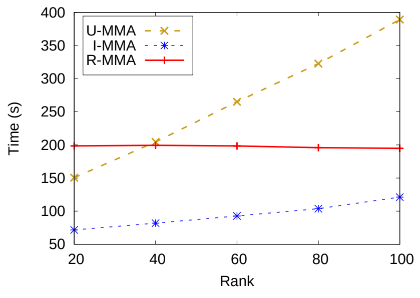

Fig. 7: Computation time comparison for learning the mixture model distributions in U-MMA, I-MMA and R-MMA with different ranks on the MovieLens 1M dataset.

## 6.2 Effectiveness of Pre-training

Since the optimization objective of MMA is non-convex and the number of parameters is larger than that of stand-alone MA method, it is non-trivial to learn accurate MMA models, e.g., the learning process will be easily trapped by local minimum. Figure 5 shows the training and test RMSEs of 1) a global mixture model with all components trained in an end-to-end way and 2) a biased MA model for 4-star ratings with increasing number of epochs. In this experiment, we learn the global MMA model by directly minimizing Equation 7 using SGD on MovieLens 1M dataset with rank = 20, and then show the training and test RMSE trends. For the individual model, we learn a biased MA model for 4-star ratings based on the proposed weighted matrix approximation method, and show the training and test RMSE trends.

As shown in Figure 5, the global model converges when test RMSE reaches 0.8545. However, the individual model towards 4 star ratings converges at a lower test RMSE — 0.8440, which confirms that the global model learning is indeed trapped by a bad local minimum. In addition, if we first pretrain the biased MA models and then learn the mixture distribution in MMA, the final test RMSE is around 0.8303 (as shown by the dotted black line). Therefore, we can conclude that pretraining the MMA models using the proposed weighted MA method is desirable because bad local minimums can be avoided after pretraining. In addition, pretraining can be performed in parallel, i.e., different biased MA models can be trained in parallel, which can improve the scalability of model training. Therefore, we use the proposed two-phase training mechanism in MMA rather than an end-to-end way in existing neural matrix factorization methods [16].

## 6.3 Comparison of MMA Variants

The proposed MMA method has three variants: 1) the user-based MMA (U-MMA), 2) the item-based MMA (I-MMA) and 3) the rating-based MMA (R-MMA) due to different mixture model assumptions. Here, we compare the accuracy and efficiency of the three variants on the MovieLens 1M dataset.

### 6.3.1 *Accuracy*

Figure 6 compares the accuracy of U-MMA, I-MMA and R-MMA with rank ranging from 20 to 100 on the MovieLens 1M dataset. As shown in the figure, I-MMA significantly outperforms U-MMA with all rank values, which is because the true rating distribution of each user is harder to estimate than that of each item due to less ratings from the users. Meanwhile, we can also see that R-MMA significantly outperforms both U-MMA and I-MMA with all rank values. The main reason is that U-MMA/I-MMA uses the same mixture distribution for all ratings from the same user/item, but R-MMA can adaptively change the distribution based on the characteristics of the users and the items. For instance, users could be very confirmative on some items, e.g., highly positive/negative ratings, but be very variable on other items, e.g., neutral ratings [2], so that it is desirable to use different mixture distributions to characterize the differences.

### 6.3.2 *Efficiency*

Figure 7 compares the learning time of the mixture model distributions for U-MMA, I-MMA and R-MMA with rank ranging from 20 to 100 on the MovieLens 1M dataset. As shown in Figure 7, I-MMA requires much less learning time than U-MMA although both methods use SGD-based learning. This is because the number of items is smaller than the number of users in the dataset. And it is not surprising to see that the time of learning the DNN models in R-MMA do not change much when rank increases, which is because the numbers of training examples are fixed for all ranks. In addition, we only use a single thread to learn the DNN models here, which means the efficiency of R-MMA can be further improved if we use parallel learning for the DNN.

In summary, R-MMA is more desirable than U-MMA and I-MMA due to higher accuracy and scalability. Therefore, the rest of the experiments will be focused on R-MMA.

## 6.4 Analysis of the Classification-based Learning

Here, we first analyze the sensitivity of R-MMA with different $\alpha$ values, classification methods to learn $\Pi$, shapes of the neural networks, weight functions in learning biased MA models and surrogate functions for the optimization objective of top-N recommendation. Then, we analyze the

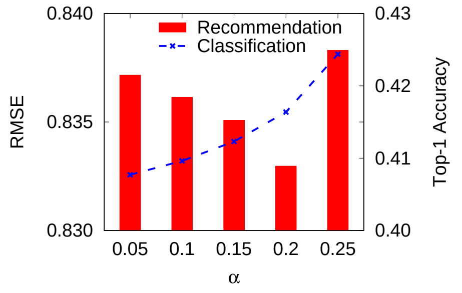

Fig. 8: Sensitivity analysis with $\alpha$ on the MovieLens 1M dataset. The left $y$-axis shows the RMSE, and the right $y$-axis shows the classification accuracy of the DNN.

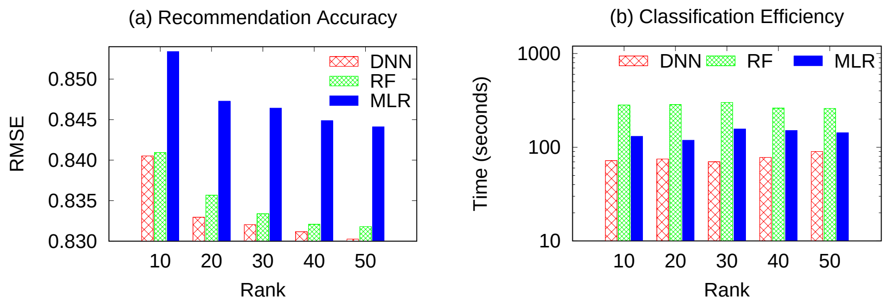

Fig. 9: Sensitivity analysis with different classification algorithms on the MovieLens 1M dataset. We use 4 threads in parallel to learn all classification models. For random forest (RF) [8], we use 100 trees in total. For multinomial logistic regression (MLR) [7], we use the iteratively reweighted least square method to improve convergence.

scalability of R-MMA. Note that the classification methods in the experiments are implemented based on the JSAT library [36].

### 6.4.1 Sensitivity with $\alpha$

Figure 8 analyzes the sensitivity of R-MMA with $\alpha$ in the weight function (Equation 9). From Equation 9, we know that the learned biased MA models become more biased when we increase $\alpha$. As shown in Figure 8, the classification accuracy of the DNN method increases when $\alpha$ increases, which is because the biased MA models are more accurate to predict the ratings on their targeted stars with large $\alpha$. Similarly, the recommendation accuracy of R-MMA also increases when $\alpha$ increases from 0.05 to 0.2. However, when $\alpha = 0.25$, the recommendation accuracy of R-MMA decreases significantly, which is because the learned MA models are too biased and the recommendation accuracy on untargeted stars are sacrificed significantly. For instance, the weight of 5-star ratings is only $\beta$ when we learn biased MA model for 1-star ratings with $\alpha = 0.25$, and vice versa.

### 6.4.2 Sensitivity with Classifiers

The classification algorithms also have impacts on the performance of R-MMA. Here, we compare DNN with two other popular methods in multi-class classification problems: Random Forest (RF) [8] and Multinomial Logistic Regression (MLR) [7]. Figure 9 (a) shows the recommendation accuracy of the three different classification algorithms in R-MMA. As shown in the results, the recommendation accuracy of DNN and RF are comparable, and DNN-based method shows slightly higher accuracy when rank $k$ increases from 10 to 50. However, the recommendation accuracy of MLR is much lower than the other two methods, which is because the targeted classification task is not suitable for linear classifiers. Therefore, non-linear classifiers, e.g., DNN and RF, are more appropriate for the proposed method.

We also compare the efficiency of the three classification methods in Figure 9 (b). As shown in the figure, DNN requires much lower learning time than RF with the same computation configurations. This indicates that DNN is more appropriate for the proposed method due to comparable accuracy and higher efficiency.

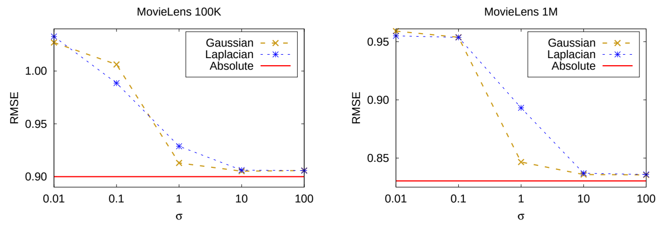

Fig. 10: Sensitivity of R-MMA with three different distance measures in the weight function (Equation 9): Gaussian kernel distance, Laplacian kernel distance and absolute distance with rank = 20 on the MovieLens 100K and 1M datasets.

### 6.4.3 Sensitivity with Neural Network Structure

Figure 12 shows the recommendation accuracy and efficiency of the proposed R-MMA method with different shapes of DNNs. As shown in Figure 12 (a), the recommendation accuracy are comparable among three different networks, deep network shows slightly better accuracy when the rank $r$ is greater than 30 and wide network shows slightly better accuracy when the rank $r$ is smaller than 30. The accuracy of the biased MA models are low when the rank is small, so that more complex network, i.e., wide network containing more edges after dropout, can achieve better performance. When the rank is large, the biased MA models can achieve good performance, so that simple network, i.e., deep network containing less edges after dropout, can achieve better generalization performance. Meanwhile, Figure 12 (b) compares the learning time of the three networks, and we can see that wide network requires much higher computation time than the other two networks due to more parameters to learn. Therefore, the deep network structure is more desirable due to higher efficiency and better accuracy when $r$ is large, because large ranks are usually adopted to achieve better recommendation accuracy in practice.

### 6.4.4 Sensitivity with Weight Function

In the proposed weighted matrix approximation method, we use absolute function to measure the distance between targeted rating of the biased model and the training rating. Here, we replace the absolute function with Gaussian kernel ($\exp\{-(x-y)^2/2\sigma^2\}$) and Laplacian kernel

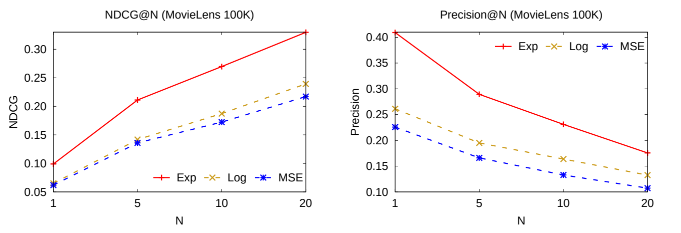

Fig. 11: Sensitivity of R-MMA with three different surrogate loss functions: exponential loss, log loss and mean square error on the MovieLens 100K dataset.

($\exp\{-|x-y|/\sigma\}$) and compare the recommendation accuracy on the MovieLens 100K and 1M datasets. As shown in Figure 10, both Gaussian and Laplacian kernel distances are very sensitive to hyperparameters, i.e., the variance of the distribution, and their performances increase when $\sigma$ increases. However, the effects of rating distance will be eliminated if $\sigma$ is too large, i.e., both Gaussian and Laplacian kernel distances become very close to a constant. Therefore, we can conclude that absolute distance is more desirable than Gaussian and Laplacian kernels due to the following two reasons: 1) no hyperparameters in the distance function and 2) the distance between the targeted rating of the biased model and the training example will not be eliminated, which can help to improve accuracy as demonstrated in the experiments. Therefore, we keep the absolute distance in the weight function for all experiments.

#### 6.4.5 *Sensitivity with Surrogate Function*

Figure 11 analyzes the sensitivity of R-MMA with three different surrogate loss functions: exponential loss (Exp), log loss (Log) and mean square error (MSE). As we can see from the results, exponential loss achieves much higher accuracy in terms of both NDCG@N and Precision@N than Log loss and mean square error. Since MSE cannot directly reflect ranking accuracy [30], using MSE as surrogate loss function cannot achieve optimal top-N recommendation accuracy. Exponential loss and log loss can both reflect ranking accuracy, but exponential loss is more accurate which may be due to its close relationship with the popular softmax loss in neural networks. Similar results can be observed from other datasets, so that we choose exponential loss as the surrogate loss function in MMA for top-N recommendations.

#### 6.4.6 *Scalability Analysis*

Figure 13 analyzes the efficiency of R-MMA with different numbers of computational threads in parallel on the MovieLens 1M dataset. Details of the parallel learning can be found in Section 4.4. As shown in the results, the efficiency of R-MMA can be significantly improved by increasing the number of threads, e.g., about 2.5X speedup when we use 5 threads compared with 1 thread. Note that 5X speedup cannot be achieved with 5 threads because the learning time of the biased MA models are different from each other. The efficiency of learning biased MA models cannot be easily improved by increasing the number of threads after 5, because we do not use parallel mechanism to learn individual biased MA models. But the efficiency of learning the DNN models can be easily improved by increasing the number of threads due to better scalability of DNN. Our empirical analysis shows that using 8 threads in R-MMA can obtain about 3X speedup.

TABLE 1: Mean average error (MAE) comparison between R-MMA and categorical matrix completion (CMC) method [9] on the MovieLens 100K dataset. We adopt the same experiment setting and compare with the result reported in the paper.

| | MAE | Improvement |
|---|---|---|
| CMC | 0.708 | - |
| R-MMA (r = 10) | $0.6974 \pm 0.0061$ | 1.49% |
| R-MMA (r = 20) | $0.6932 \pm 0.0065$ | 2.08% |
| R-MMA (r = 50) | $0.6902 \pm 0.0059$ | 2.51% |

TABLE 2: RMSE comparison between R-MMA and the matrix completion from quantized measurements (MCQM) method [5] on the MovieLens 1M dataset. We adopt the same experiment setting and compare with the result reported in the paper.

| | RMSE | Improvement |
|---|---|---|
| MCQM | $0.8568 \pm 0.0014$ | - |
| R-MMA (r = 7) | $0.8481 \pm 0.0015$ | 1.01% |

### 6.5 Accuracy Comparison

#### 6.5.1 *Comparison with Categorical Matrix Approximation Methods*

Table 1 compares the recommendation MAE between R-MMA and one of the state-of-the-art categorical matrix approximation methods — CMC [9] on the MovieLens 100K dataset. Following the settings in their paper [9], we split the dataset into training and test sets by 95%:5% and report the test MAE by averaging over five random splits. Since the rank in their experiment is not mentioned, we consider ranks from 10 to 50 in the R-MMA method. As we can see from the results, R-MMA significantly outperforms CMC with all ranks, and the relative improvements range from 1.49% to 2.51% when the rank increases from 10 to 50.

Table 2 compares the recommendation RMSE between R-MMA and another state-of-the-art categorical matrix approximation method — MCQM [5] on the MovieLens 1M dataset. Following the setting in the original paper [5], we randomly split the dataset into training and test sets by 80%:20%, set the rank to 7, and report the test RMSEs by averaging over 20 random splits. The results show that R-MMA can achieve significantly lower test RMSE compared with MCQM and the relative improvement is about 1.01%.

The above results (Table 1 and Table 2) show that the R-MMA method can achieve higher accuracy than state-of-the-art categorical matrix approximation methods on rating prediction task. The main reason is that MMA uses biased matrix approximations as the link functions to transform the predicted ratings. However, the CMC and MCQM methods both adopt logistic regression methods as the link function, which should not be as powerful as matrix approximation models in collaborative filtering.

#### 6.5.2 *Comparison with State-of-the-art Methods*

Table 3 compares the recommendation accuracy between the proposed R-MMA method and six state-of-the-art matrix

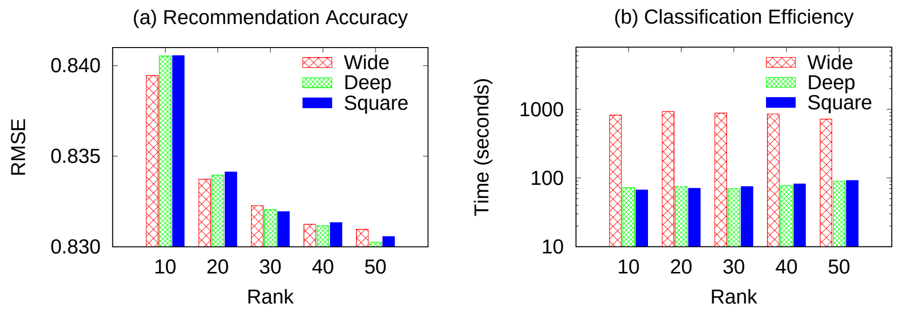

Fig. 12: Sensitivity analysis with the DNN structure on the MovieLens 1M dataset. Here, we compare the accuracy and efficiency of three different structures: wide network (2 × 512), deep network (64 × 16), and square network (32 × 32).

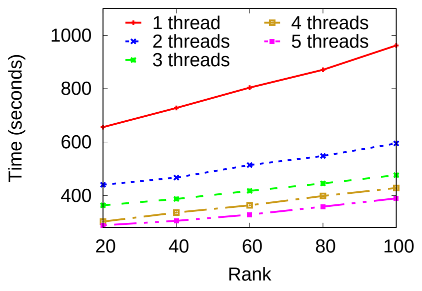

Fig. 13: Efficiency analysis of R-MMA with different computational threads on the MovieLens 1M dataset.

TABLE 3: RMSE comparison between R-MMA and six state-of-the-art matrix approximation-based collaborative filtering algorithms [11], [26], [27], [29], [32], [48] on the MovieLens (10M), Netflix Prize and Amazon Instant Video datasets.

| | ML-10M | Netflix | Amazon |
|---|---|---|---|
| DFC | $0.8067 \pm 0.0002$ | $0.8453 \pm 0.0003$ | $0.9953 \pm 0.0257$ |
| GSMF | $0.8012 \pm 0.0011$ | $0.8420 \pm 0.0006$ | $1.1347 \pm 0.0223$ |
| LLORMA | $0.7855 \pm 0.0002$ | $0.8275 \pm 0.0004$ | $0.9871 \pm 0.0221$ |
| WEMAREC | $0.7775 \pm 0.0007$ | $0.8143 \pm 0.0001$ | $1.0093 \pm 0.0219$ |
| SMA | $0.7682 \pm 0.0003$ | $0.8036 \pm 0.0004$ | $0.9869 \pm 0.0237$ |
| MRMA | $0.7634 \pm 0.0009$ | $0.7973 \pm 0.0002$ | $0.9798 \pm 0.0236$ |
| **R-MMA** | $\mathbf{0.7606 \pm 0.0002}$ | $\mathbf{0.7948 \pm 0.0001}$ | $\mathbf{0.9682 \pm 0.0284}$ |

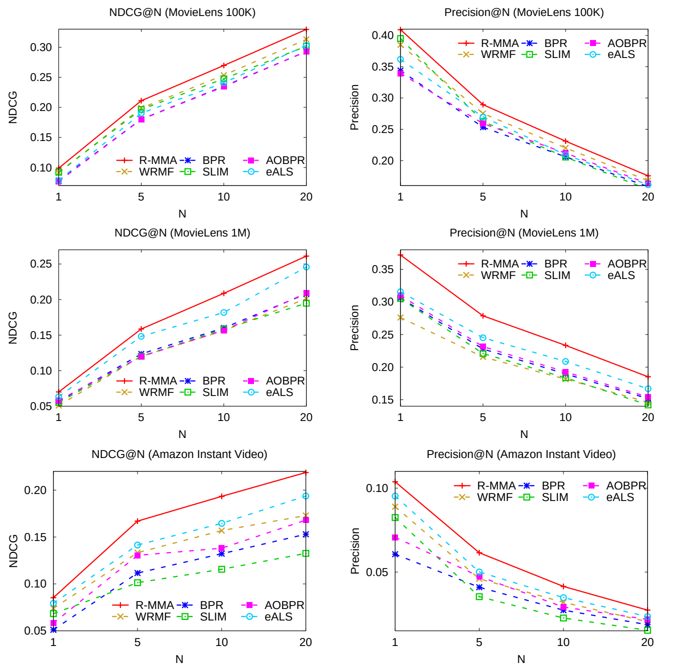

Fig. 14: NDCG@N and Precision@N comparison between R-MMA and five state-of-the-art MA-based top-N recommendation methods on the MovieLens 100K, MovieLens 1M and Amazon Instant Video datasets.

approximation-based methods [11], [26], [27], [29], [32], [48]. Among the compared methods, DFC [32], LLORMA [26], WEMAREC [11] and MRMA [27] are ensemble MA methods, in which multiple biased MA models are also used in their recommendation steps. As shown in the results, the proposed R-MMA method statistically significantly outperforms all the compared methods on MovieLens 10M, Netflix and Amazon datasets. Compared with the most recent work — MRMA, the proposed method can achieve 1.2% relative improvement on the Amazon dataset, i.e., the improvement over MRMA should be considered as significant. Note that the hyperparameters of R-MMA is only tuned on MovieLens 1M dataset and then fixed for all comparisons but the compared methods (e.g., MRMA) were tuned on the MovieLens 10M and Netflix datasets directly, so that larger improvements over the compared methods could be obtained if we tune the hyperparameters for each dataset individually. Figure 14 compares the recommendation accuracy between R-MMA and five state-of-the-art MA-based top-N recommendation methods [17], [19], [33], [37], [38]. As shown in the results, R-MMA statistically significantly outperforms all the compared methods on MovieLens 100K, MovieLens 1M and Amazon datasets with both NDCG@N and Precision@N when N increases from 1 to 20.

The main reasons why the proposed method can achieve higher accuracy are: 1) mixture model assumptions over the user-item ratings are more appropriate in collaborative filtering tasks; 2) the proposed method can more accurately characterize the diverse user/item interests by learning multiple biased matrix approximation models and 3) the mixture model in MMA can be regarded as an ensemble method which combines $K$ different biased models in recommendation, which can naturally reduce the estimation variance [24], [50].

## 7 Related Work

Collaborative filtering methods are important in real-world recommender systems [1], [4], [13], [31]. Among existing CF algorithms, matrix approximation-based methods have achieved great success in both rating prediction [3], [23], [26], [29], [41], [49] and top-N recommendation [19], [38], [47]. Since the user-item rating matrices in real-world recommender systems are typically very sparse, MA-based methods which can address the data sparsity issue by reducing the dimensionality of user/item feature vectors can outperform the classic memory-based CF methods in terms of accuracy [6], [23], [42]. To further improve the performance, different variants of matrix approximation methods, e.g., the PMF method [41], the BPMF method [40], the SVD++ method [22], the GSMF method [48], the SMA method [29], etc., have been proposed to improve model learning [29], [40], [41], [48] or incorporate more user-item interactions [22]. Besides rating prediction task, some other works tried to tackle the implicit feedback issue in top-N recommendation by introducing different weights between observed ratings and missing ratings [17], [19], [34] and solving pair-wise loss functions during model learning [37], [38].

Mixture or ensemble matrix approximation methods have also been proposed recently to achieve higher accuracy [10], [26], [27] and/or better scalability [11], [32]. The Divide-Factor-Conquer (DFC) framework [32] was proposed to address the scalability issue of large-scale matrix factorization, in which they divide the matrix into sub-matrices, then factorize each sub-matrix in parallel, and finally combine all sub models in prediction. The LLORMA method [26] assumed the locally low-rank structure in the rating matrix, in which they choose sub-matrices by kernel distances to some anchors, then factorize each sub-matrix using weighted matrix approximation, and finally integrate the sub-matrix approximations using kernel smoothing. The WEMAREC method [11] proposed a co-clustering-based sub-matrices generation method. Then, they factorize each sub-matrix using weighted matrix approximation. Finally, all sub-matrix approximations are combined using weighted average. Besides scalability, accuracy can also be boosted by ensemble/mixture models. Chen et al. proposed the MPMA method [10], in which a mixture of global and local matrix approximation models is trained to capture the global interests and unique interests of users and items. Recently, Li et al. proposed the MRMA method [27], in which they assume that mixture low-rank structures exist in real-world rating matrices and they proposed to combine the approximations from matrix approximation models with different ranks to improve the recommendation accuracy.

Categorical matrix approximation methods have been recently proposed to address the matrix approximation problem on categorical/quantized data [5], [9]. Cao and Xie [9] proposed to consider the problem of completing a matrix with categorical-valued entries from partial observations. They proposed to recover the targeted matrix by maximizing the likelihood ratio with nuclear norm constraint and mapping the entries through multinomial logistic regression link functions. Later, Bhaskar [5] considered the recovery of a low rank real-valued matrix given a subset of noisy and quantized measurements, in which maximum likelihood estimation is considered under a constraint on the entry-wise infinity-norm and an exact rank constraint. Also, this method can achieve faster convergence rate when the fraction of revealed observations is fixed.

Compared with the above works, the proposed method mainly differs in the following two aspects. 1) Higher accuracy. As demonstrated in the experiments, R-MMA statistically significantly outperforms two categorical matrix approximation methods and eleven state-of-the-art matrix approximation-based collaborative filtering methods in both rating prediction and top-N recommendation tasks. 2) Decent scalability. The biased matrix approximation models in the proposed method can be learned in parallel, which can significantly reduce the overall learning time. Meanwhile, the learning of the mixture distributions can also be highly scalable if we choose scalable classification methods, e.g., DNN. In addition, the proposed method requires much lower overall computations compared with many existing ensemble/mixture MA methods [11], [26], [27], [32] due to much less number of base models.

## 8 Conclusion

The user-item ratings are typically quantized in collaborative filtering tasks, so that different users can have diverse interests on the same item. Many existing matrix approximation methods, which learn global user/item feature vectors, cannot accurately characterize the diverse interests of users on the items and thus achieve suboptimal recommendation accuracy. To this end, this paper proposes a mixture matrix approximation method, in which we learn different user/item feature vectors for different stars to better characterize the diverse interests of users/items. Empirical studies on real-world datasets demonstrate that the proposed method can achieve higher accuracy than state-of-the-art matrix approximation-based collaborative filtering methods in both rating prediction and top-N recommendation and meanwhile achieve high scalability.

## Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61932007 and U1630115.

## References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[2] X. Amatriain, J. M. Pujol, and N. Oliver. I like it... i like it not: Evaluating user ratings noise in recommender systems. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization*, UMAP '09, pages 247–258. Springer, 2009.

[3] A. Beutel, A. Ahmed, and A. J. Smola. Accams: Additive co-clustering to approximate matrices succinctly. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 119–129, 2015.

[4] A. Beutel, E. H. Chi, Z. Cheng, H. Pham, and J. Anderson. Beyond globally optimal: Focused learning for improved recommendations. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 203–212, 2017.

[5] S. A. Bhaskar. Probabilistic low-rank matrix completion from quantized measurements. *The Journal of Machine Learning Research*, 17(1):2131–2164, 2016.

[6] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, 1998.

[7] D. Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1):197–200, 1992.

[8] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[9] Y. Cao and Y. Xie. Categorical matrix completion. In *The 2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 369–372. IEEE, 2015.

[10] C. Chen, D. Li, Q. Lv, J. Yan, S. M. Chu, and L. Shang. MPMA: mixture probabilistic matrix approximation for collaborative filtering. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI '16)*, pages 1382–1388, 2016.

[11] C. Chen, D. Li, Y. Zhao, Q. Lv, and L. Shang. WEMAREC: Accurate and scalable recommendation through weighted and ensemble matrix approximation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*, pages 303–312, 2015.

[12] G. E. Dahl, T. N. Sainath, and G. E. Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *The 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '13)*, pages 8609–8613. IEEE, 2013.

[13] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, pages 271–280. ACM, 2007.

[14] D. Dueck and B. Frey. Probabilistic sparse matrix factorization. *University of Toronto technical report PSI-2004-23*, 2004.

[15] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[16] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. 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.

[17] X. He, H. Zhang, M.-Y. Kan, and T.-S. 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*, SIGIR '16, pages 549–558. ACM, 2016.

[18] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[19] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, 2008.

[20] S. Kabbur, X. Ning, and G. Karypis. Fism: Factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 659–667. ACM, 2013.

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

[22] Y. 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 (KDD '14)*, pages 426–434. ACM, 2008.

[23] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[24] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'94, pages 231–238. MIT Press, 1994.

[25] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 85–96. ACM, 2014.

[26] J. Lee, S. Kim, G. Lebanon, and Y. Singer. Local low-rank matrix approximation. In *Proceedings of The 30th International Conference on Machine Learning (ICML '13)*, pages 82–90, 2013.

[27] D. Li, C. Chen, W. Liu, T. Lu, N. Gu, and S. Chu. Mixture-rank matrix approximation for collaborative filtering. In *Advances in Neural Information Processing Systems 30*, pages 477–485, 2017.

[28] D. Li, C. Chen, Q. Lv, H. Gu, T. Lu, L. Shang, N. Gu, and S. M. Chu. Adaerror: An adaptive learning rate method for matrix approximation-based collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 741–751. ACM, 2018.

[29] D. Li, C. Chen, Q. Lv, J. Yan, L. Shang, and S. Chu. Low-rank matrix approximation with stability. In *The 33rd International Conference on Machine Learning (ICML '16)*, pages 295–303, 2016.

[30] P. Li, Q. Wu, and C. J. Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2008.

[31] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[32] L. W. Mackey, M. I. Jordan, and A. Talwalkar. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1134–1142, 2011.

[33] X. Ning and G. Karypis. SLIM: Sparse linear methods for top-n recommender systems. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 497–506, 2011.

[34] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 502–511. IEEE Computer Society, 2008.

[35] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD CUP'07*, pages 5–8, 2007.

[36] E. Raff. Jsat: Java statistical analysis tool, a library for machine learning. *Journal of Machine Learning Research*, 18(23):1–5, 2017.

[37] S. Rendle and C. Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 273–282, 2014.

[38] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461, 2009.

[39] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 251–258. ACM, 2008.

[40] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning (ICML '08)*, pages 880–887. ACM, 2008.

[41] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems (NIPS '08)*, pages 1257–1264, 2008.

[42] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system - a case study. In *ACM WebKDD '2000 Workshop*. ACM, 2000.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[44] P. Symeonidis and A. Zioupos. *Matrix and Tensor Factorization Techniques for Recommender Systems*, volume 1. Springer, 2016.

[45] J. Wang, P. Zhao, and S. C. Hoi. Cost-sensitive online classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2425–2438, 2013.

[46] J. Wasilewski and N. Hurley. How diverse is your audience? exploring consumer diversity in recommender systems. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, 2017.

[47] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. COFIRANK maximum margin matrix factorization for collaborative ranking. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1593–1600, 2007.

[48] T. Yuan, J. Cheng, X. Zhang, S. Qiu, and H. Lu. Recommendation by mining multiple user behaviors with group sparsity. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI '14)*, pages 222–228, 2014.

[49] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1511–1520. ACM, 2013.

[50] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

**Dongsheng Li** is a research staff member with IBM Research – China since April 2015. He is also an adjunct professor with School of Computer Science, Fudan University, Shanghai, China. He obtained Ph.D. from School of Computer Science of Fudan University, China, in 2012. His research interests include recommender systems and general machine learning applications. In April 2018, he won one of the highest technical awards in IBM – the IBM Corporate Award. He is a member of IEEE.

**Chao Chen** is a PhD candidate in school of Electronic Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai, China. He joined IBM Research - China since May 2016, and before that he received his Master degree in Computer Science from Tongji University, Shanghai, China in 2016. His research interests focus on applying neural networks techniques to recommender systems.

**Tun Lu** is an Associate Professor in School of Computer Science at Fudan University. He earned his Ph.D. in Computer Science from Sichuan University in 2006 and was a visiting scholar in HCI Institute at Carnegie Mellon University in 2015. His research interests include Computer Supported Cooperative Work (CSCW), Social Computing, and Human-Computer Interaction (HCI). He has been active in professional services by serving as PC Co-chairs (e.g. ChineseCSCW'17 & 18, CSCWD'10), Associate Chairs (e.g. CHI'19 & 20, CSCW'19), PC members (e.g. GROUP'18, CRIWG'17 & 2018, CSCWD'16), Guest Editors (e.g. Int. J. Coop. Inf. Syst.), and reviwers for many well-known journals and conferences. He has published over 60 peer-reviewed publications in prestigious journals and conferences such as CSCW, CHI, WWW, NIPS, UbiComp, etc. He shared a Best Paper award at CSCW'15 and an Honorable Mention award at CSCW'18. He is a senior member of China Computer Federation (CCF), a member of ACM and IEEE. He is a standing committee member of CCF Technical Committee of Cooperative Computing.

**Stephen M. Chu** studied physics at Peking University in Beijing, China, and received the MS and PhD degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign. He is a senior technical staff member at IBM. His team received the top prize in the DARPA-organized speech recognition contest for five consecutive times. His pioneering work in Audio-Visual Speech recognition was featured in the PBS program Scientific American Frontiers with Alan Alda. His research interests cover speech recognition, computer vision, and signal processing.

**Ning Gu** is a full time professor in school of computer science at Fudan University. His research interests include data consistency, data analysis and recommendation techniques, social and technical combination research on specific domain, currently he is more interested in two application domains on the disabled/seniors and saving energy. Prof.Gu works in the area of Computer Supported Cooperative Work (CSCW) and Social Computing, and Human Computer Interaction (HCI). He has published over 130 papers in the key international conferences and journals such as CHI, CSCW, WWW, NIPS, TPDS. Prof. Gu is an IET fellow, director of Fudan Social Computing Research Center, director of Cooperative Information and Systems Laboratory. He also serves as director of the Technical Committee on Cooperative Computing of China Computer Federation (CCF), director of the Technical Committee on Cooperative Computing of Shanghai Computer Society.