Differentiating Regularization Weights – A Simple Mechanism to Alleviate Cold Start in Recommender Systems

HUNG-HSUAN CHEN and PU CHEN, National Central University

Matrix factorization (MF) and its extended methodologies have been studied extensively in the community of recommender systems in the last decade. Essentially, MF attempts to search for low-ranked matrices that can (1) best approximate the known rating scores, and (2) maintain low Frobenius norm for the low-ranked matrices to prevent overfitting. Since the two objectives conflict with each other, the common practice is to assign the relative importance weights as the hyper-parameters to these objectives. The two low-ranked matrices returned by MF are often interpreted as the latent factors of a user and the latent factors of an item that would affect the rating of the user on the item. As a result, it is typical that, in the loss function, we assign a regularization weight λ_p on the norms of the latent factors for all users, and another regularization weight λ_q on the norms of the latent factors for all the items. We argue that such a methodology probably over-simplifies the scenario. Alternatively, we probably should assign lower constraints to the latent factors associated with the items or users that reveal more information, and set higher constraints to the others. In this article, we systematically study this topic. We found that such a simple technique can improve the prediction results of the MF-based approaches based on several public datasets. Specifically, we applied the proposed methodology on three baseline models – SVD, SVD++, and the NMF models. We found that this technique improves the prediction accuracy for all these baseline models. Perhaps more importantly, this technique better predicts the ratings on the long-tail items, i.e., the items that were rated/viewed/purchased by few users. This suggests that this approach may partially remedy the cold-start issue. The proposed method is very general and can be easily applied on various recommendation models, such as Factorization Machines, Field-aware Factorization Machines, Factorizing Personalized Markov Chains, Prod2Vec, Behavior2Vec, and so on. We release the code for reproducibility. We implemented a Python package that integrates the proposed regularization technique with the SVD, SVD++, and the NMF model. The package can be accessed at https://github.com/ncu-dart/rdf.

CCS Concepts: • **Information systems** → **Collaborative filtering**; *Computational advertising*; *Online advertising*; • **Applied computing** → **Electronic commerce**; *Online shopping*; • **Computing methodologies** → *Machine learning*; *Factorization methods*;

Additional Key Words and Phrases: Recommender systems, matrix factorization, collaborative filtering, SVD, SVD++, cold start, long tail

ACM Reference format:
Hung-Hsuan Chen and Pu Chen. 2019. Differentiating Regularization Weights – A Simple Mechanism to Alleviate Cold Start in Recommender Systems. *ACM Trans. Knowl. Discov. Data* 13, 1, Article 8 (January 2019), 22 pages.
https://doi.org/10.1145/3285954

We acknowledge partial support by the Ministry of Science and Technology (MOST 107-2221-E-008-077-MY3), the Industrial Technology Research Institute (ITRI 107-W1-21A2), and CHANGING.AI (https://www.changing.ai/).

Authors' addresses: H.-H. Chen and P. Chen, Computer Science and Information Engineering, National Central University, No. 300, Zhongda Rd., Zhongli District, Taoyuan City 32001, Taiwan (R.O.C.); emails: hhchen@ncu.edu.tw, greazepasta@gmail.com.

1556-4681/2019/01-ART8 $15.00
https://doi.org/10.1145/3285954

1 INTRODUCTION

Knowledge is power. The formation of knowledge requires human to analyze the available information, discover the patterns from the analysis, and generalize the common rules. However, when the number of information explodes, the large amount of information with low relevance may become the burden for users to select, organize, and discover the useful information. This phenomenon is called *information overload* [14]. Studies have shown that perceived information overload may increase a user's satisfaction but would lower the quality of the decisions [43].

Internet is the main channel for information sharing nowadays. However, Internet is also the main cause of information overload. As a result, we need various tools, such as the search engines and the recommender systems, to effectively filter most information and preserve few information that best fit a user's requirement. As long as the application domain contains the information that beyond an individual's capacity to digest in a short period, we probably should introduce recommender systems to filter most information. Indeed, recommender systems are applied in various domains, including e-commerce, education, academic research, financing, fashion, online dating, and many more [2, 8, 10, 11, 17, 29, 33, 35].

Most algorithms used by the recommender systems fall into one of the following three types: collaborative filtering (CF), content-based, or hybrid approach. Among all the algorithms, matrix factorization (MF) and its extensions are probably the most famous type. MF is widely studied in the last decade, probably because of its success in the Netflix challenge,[1] which targets to tackle the rating prediction task, i.e., predicting a user i's rating r_{ij} on an unrated item j. Essentially, MF compiles a rating matrix $\boldsymbol{R} = [r_{ij}]$ to record all the known ratings and attempts to decompose the rating matrix as the product of two low-ranked matrices, which usually referred as the latent factors of the users and the latent factors of the items respectively. The decomposition can be modeled as an optimization problem to minimize the loss function, which is typically composed by the following two parts: (1) training error – the root-mean-squared error (RMSE) between the known ratings and the predicted ratings, and (2) the Frobenius norms of the two low-ranked matrices, which used to estimate the inverse of the generalization power to the unseen data. The optimization strategy is generally based on stochastic gradient descent (SGD), alternating least squares (ALS), or their variations, such as ADAM, Nesterov, RMSprop, and so on.

Although various machine learning tasks adopt similar loss functions, we found a key difference between the rating prediction task and most of the other machine learning tasks. In particular, for most machine learning tasks, we may obtain more clues about the relationship between the target variables and the features, once we collect more training data. However, if we apply the MF technique for the rating predicting task, as we collect more training data, we probably have more information of certain items/users (especially the popular items and the heavy users), but not necessarily the others (e.g., the long tail items and the less active users). As a result, assigning the same regularization λ_q to the latent factors for all items and the same regularization λ_p to the latent factors for all users, as most studies reported, are probably too naïve.

We argue that, we should assign lower regularization weights to the latent factors associated with the items that received many ratings, because we get more clues of these items from many users' collective ratings. Similarly, as a user rated more items, this user may reveal more information about her/his taste, so we should assign a lower weight to the regularization term associated with the latent factors of this user. In this article, we systematically study this effect. We found that this simple technique is effective for the MF-based approach on most of the tested datasets. More importantly, this technique can make good predictions on the less active user's ratings on the

[1] https://www.netflixprize.com/.

long tail items. As a result, our proposed method may partially alleviate the cold-start problem. Additionally, this technique is simple and general enough to be applied to many other recommendation models, such as Factorizing Personalized Markov Chains (FPMC) [50], Prod2Vec [18], Behavior2Vec [7], Factorization Machines (FM) [49], Field-aware Factorization Machines (FFM) [26], WSVD [6], and neural CF [23], that target at the rating predicting task or the recommendation tasks.

The article makes the following contributions.

- Although the MF-based methods (e.g., SVD[2] [28], SVD++ [28], and NMF [16]) are widely used by many recommender systems, we argue that these methods may not fully utilize the information revealed by the popular items and the active users. We conducted experiments on public datasets to validate such an argument.
- We extended the SVD, SVD++, and the NMF models based on three simple *regularization differentiating function*s (RDF) so that the regularization weights on different items and users are different. Particularly, these functions set weaker constraints on the popular items and the active users and stronger constraints on the long tail items and the less active users. As a result, the models may utilize more information revealed by the popular items and the active users, and make conservative predictions on the long tail items and the less active users.
- We empirically validated the effectiveness of the proposed method and the RDF on several public datasets. The results showed that, by integrating the RDF to the MF-based approaches, we can better predict users' preferences on items.
- We open sourced our code for public use and for reproducibility.

The rest of the article is organized as follows. Section 2 reviews the related work. In Section 3, we show the observations on several public datasets to support our claim – conventional recommender systems may have more clues on the popular items and the active users, so the conventional MF-based method can better predict an active user's ratings on the popular items. In Section 4, we propose three functions to differentiate the regularization weights on different users and different items. Section 5 shows the experimental results on the public datasets. The MF-based approaches, when accompany with the proposed functions, better predicts users' ratings on items in most cases. Finally, we discuss our discoveries and future research directions in Section 6.

2 RELATED WORK

2.1 Recommender Systems

Based on the recommendation strategies, recommender systems can be categorized into three types – content-based, CF, and the hybrid approach.

The content-based methods mostly rely on the text information or the tag/attribute information of the objects (i.e., users or items) to define the distance between objects. As a result, the system can find the most similar objects to the users or to the items that are viewing or just purchased by users. The nearest objects can be used as the recommendations. This type of approach is highly influenced by the works in the information retrieval studies.

The other type of recommendation strategy – CF – leverages on many users' collective behaviors to make recommendations. One of the most influential method of CF is MF (a.k.a. the SVD model) [28], which models the interaction between the users and the items by a large matrix and

[2]Although sharing many similarities, the SVD model used in recommender systems is different from the classic SVD (Singular Value Decomposition) model in the linear algebra. For example, it is challenging to decompose a matrix with unknown entry values based on the classic SVD model in linear algebra.

attempts to decompose the large matrix into the product of small matrices. These small matrices may serve as the representative but latent features to decide users' preferences on the items. Such a strategy motivates many following works, such as the SVD++ model [28] and the NMF model [16], and the strategy has been applied to many domains beyond the recommender systems, including link prediction in the social network and graphs [9, 40], word embedding generation [31], community detection [46], and so on.

Recently, researchers have started to apply the deep learning techniques on the recommender systems to model higher degree of interactions between the latent factors of the users and the items [23, 56]. It is also found that the embeddings of the items discovered during the training process may be utilized to infer the relationship between the items, behaviors, and potentially other types of objects [7, 18]. Since users' behaviors can be modeled as a sequence, it is natural to incorporate with the Recurrent Neural Network (RNN) model to make predictions [24]. Along these line, some suggested that the latent factors of users and items may also vary over time. Therefore, the RNN model can also be used to generated the latent factors, which are further used to predict users' preferences on the items over time [55]. Most of the deep learning-based recommendation models still leverage on users' collective behaviors to predict the preferences. Therefore, they fall into the category of CF.

Since CF is independent of the text and tag information of the users and the items, it can be applied on various types of recommender systems with little customization. However, for the less active users and the long tail items, the performance of CF is dissatisfied, because the systems have limited clues on these users and items. This is called the *cold-start* problem, which is one main disadvantage of CF. Our proposed method can partially remedy the cold-start issue. Although we apply the proposed technique on the SVD, SVD++, and NMF models for the experiments, such a technique can be applied on other learning-based rating prediction models with little modification.

2.2 Cold Start Problem in Collaborative Filtering

Cold start is a main disadvantage of CF. This section reviews some of the important works that attempt to address this issue. We categorize the related works into three categories, as illustrated in the following three paragraphs.

Several studies proposed to obtain new users' preferences based on a questionnaire, which may ask users to rate on a list of items [47, 48, 53, 57]. These items can be selected in a static (i.e., a universal set of questions) or a dynamic manner, (e.g., the following questions depends on the answers of the earlier questions). The dynamic items and questions are usually generated by machine learning models, such as the decision tree classifier [53, 57]. To maintain a consistent user experience, some may choose to conduct the questionnaire in a less intrusive manner. For example, a recommender system may show the recommended items based on the exploitation-exploration strategy – recommending the high payoff items by exploiting the (limited) knowledge on the users and showing other items to explore the unknown part of the users. The multi-armed bandit algorithms study the tradeoff between the exploitation and exploration in a systematic manner [32].

Instead of conducting explicit or implicit questionnaire, some studies incorporated the affiliated information of the new users or new items to make recommendations. For example, in many cases, the items are associated with the texts, such as the item name, description, and so on. The CF may incorporte with these texts to make recommendation on the cold start items [30]. Other affiliated information that were used to help recommendation on the cold start items (or cold start users) include item category [41], actors list (when the items are movies) [51], users' demographic information [34, 44], images of the items [39], and so on.

Recently, researchers started to work on the cold start problem based on transfer learning, which makes recommendations on the cold start items or users based on the knowledge learned

elsewhere. Specifically, a cold start user of one website a could be an active user of another site b. As a result, it is possible to learn this user's preferences based on her/his activity on site b and make recommendation on site a [1, 4]. Similarly, a cold start item on one website a could be a popular item on another website b, so one can learn the knowledge of this item from site b, and apply the knowledge on a.

Our proposed method – Differentiating Regularization Weights – is different from the above methods, because the proposed method requires no questionnaire, no affiliated information of users or items, and no related datasets to learn and transfer the knowledge. In fact, our proposed method can be naturally integrated with many of the above mentioned methods to alleviate the cold start problem altogether. For example, it is straightforward to integrate the differentiating regularization function with the objective function proposed in [57] based on each item's received number of explicit or implicit ratings and each user's number of rated or interacted items, as we will introduced later in Section 4. As a result, our proposed method is not a substitution to many of the related works mentioned above, but a general approach that can be integrated with many learning-based recommendation models to improve the recommendation quality.

2.3 Model Complexity and Regularization

The complexity of a learning model is one key factor to determine the effectiveness of the predictions. As a model becomes more complex, the model is likely to better fit the training data. Since a complex model has a larger hypothesis space, it has a higher chance to include the true relationship between the features and the target variables. However, if the available number of training instances is limited, training an (over-)complex model may cause over-fitting, i.e., obtaining the parameters that can fit the training data excellently, but may not be the good parameters for the unseen instances. This is because a complex model is sensitive to small fluctuations in the training data. Researchers call such type of error as variance. On the other hand, a simple learning model is robust to the fluctuations in the training data. However, since a simple model has a smaller hypothesis space, the true relationship between the features and the target variable may not appear in the hypothesis space. Such error is often called bias. As a result, an (over-)simple model may cause under-fitting, i.e., the model is too simple to capture the underline structure of the data.

Determining an adequate model complexity is difficult. One popular way to control model complexity is to introduce the regularization terms to smooth the best fitted curve. As a result, a small change in the training data does not make a huge fluctuation to the model, even if the model is complex. The commonest regularization term is probably the Lp norms of the parameters to learn. When we apply the linear model with the L1 norm as the regularization term, this becomes the famous Lasso method, which tends to shrink some parameters to zero and retain only the most powerful features [54]. When the regularization term is the L2 norm, this becomes the ridge regression [25]. Elastic net is another common approach that combines both the L1 norm and the L2 norm as the regularization terms. Unsurprisingly, the property of the Elastic net is a mixture of the L1 norm and the L2 norm [58]. However, the Elastic net requires to assign another hyper-parameter to determine the relative importance between the L1 norm and the L2 norm. In general, we usually control the complexity of the model by varying the value of the regularization weight. However, obtaining an adequate regularization weight usually requires extensive experiments, like cross-validation [15, 27].

Controlling the model complexity is an important issue. Here we list some other techniques to control the model complexity or prevent overfitting – (1) feature selection, which removes the less relevant features to improve the generalizability [42]; (2) early stopping, which stops the learning iterations once the error in the validation set increases [5]; (3) ensemble method, which combines

the predictions from different models to smooth out the predictions to prevent the overfitting of single model [45]; (4) dropout – dropping out some of the neural nodes during training [52].

While the relationship between model complexity and overfitting/underfitting, as well as the bias-variance tradeoff, are widely studied in the fields of machine learning and statistical learning [12, 15, 20, 42], applying the concept of regularization with the cold-start problem in the CF is rarely studied, if any. Our proposed method may bridge this gap.

3 ITEM POPULARITY, USER ACTIVITY, AND PREDICTABILITY

We argue that, if an item is rated by many users, we have more clues on the item, so we can probably discover the latent factors that better summarize the property of the item. On the other hand, we have less clues to the items that were rated by few users, so the discovered latent factors could be noisy. A similar argument may also be applied on the users – if a user interacted with (e.g., rated) more items, this user reveals more information of the personal interest. As a result, it is likely that we may discover a vector of latent factors that better represents this individual's taste. For the same reason, if a user interacted with few items, we can only make judgement based on the limited information, and thus the derived vector of latent factors is probably less convincing.

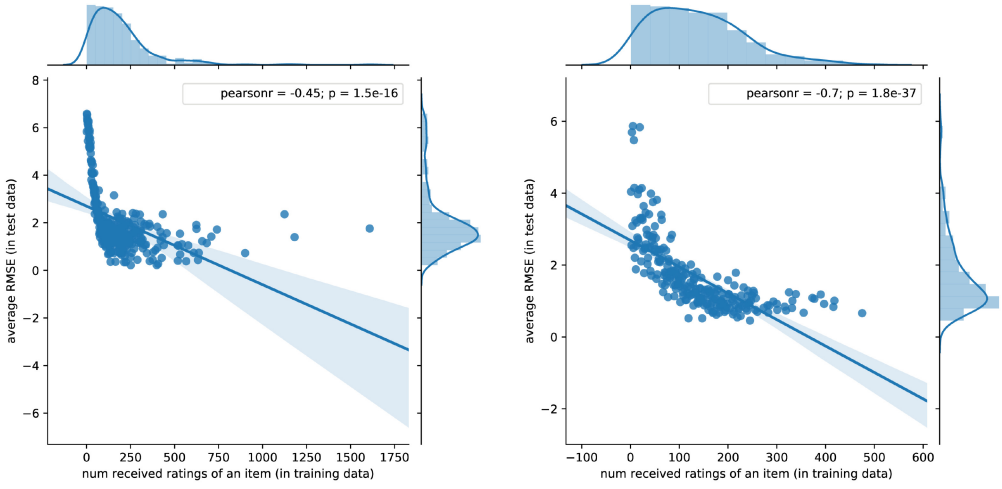
The above argument motivates this study. However, to ensure that the argument is valid, we first make an observation on two public datasets – the Epinions rating dataset and the MovieLens-100K rating dataset. We want to empirically validate that the future ratings can be better predicted if an item has received many ratings. Likewise, we also want to validate that, as users interacted with the system more often, we may obtain more information about the user, and therefore better predict personal flavor.

Specifically, we randomly split the ratings of each dataset into the training and the test datasets. We computed the number of ratings received by each item (in the training data) and the RMSE score between each item's predicted and received ratings in the test data.[3] As shown in Figure 1(a) and (b), the RMSE score decreases (i.e., better prediction) as the items received more ratings. Specifically, the Pearson correlation coefficients in both cases are less than 0, suggesting a negative correlation between the two. Additionally, such correlations are extremely significant, which can be suggested from the small p-values. We also show in the figures the confidence intervals, which are estimated by bootstrapping [13]. The observations indicate that the claim to motivate the study seems correct. Therefore, we probably should assign different constraints to the items according to the number of ratings an item had received.

We conducted similar experiments to show the relationship between each user's number of rated items (in the training data) and the RMSE score (in the test data). We plotted their relationship in Figure 2, in which each dot located at the position (x, y) denotes that the average RMSE score (on the test data) is y for all the users who rated x items (in the training data). As shown in both the Epinions dataset (Figure 2(a)) and the MovieLens-100K dataset (Figure 2(b)), the average test RMSE score decreases (i.e., better prediction) as the number of interacted items increases in the training data. The result matches our claim. Thus, it could be beneficial to assign different constraints to different users based on the number of items a user had rated.

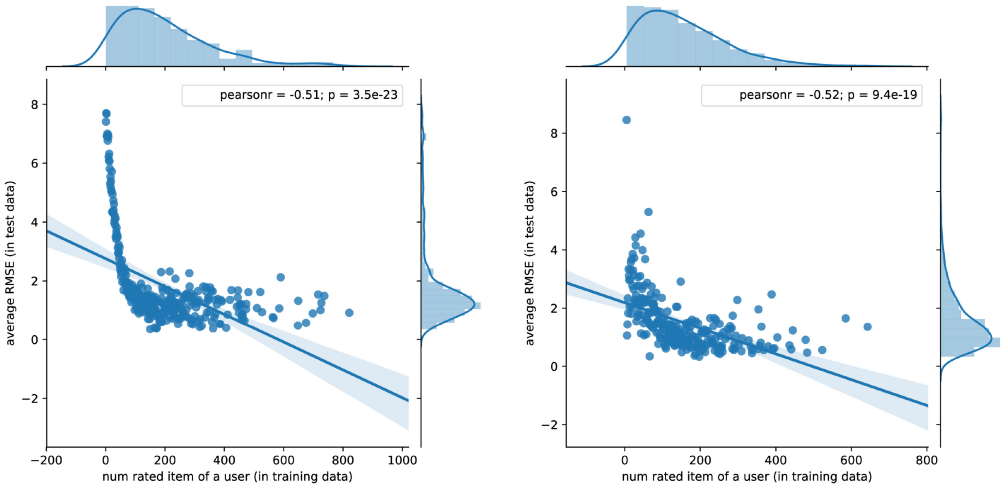
We repeated these two experiments on other three public datasets, namely the FilmTrust rating dataset, the Yahoo! Movies rating dataset, and the Amazon Musical Instruments (AMI) rating dataset. We report the Pearson correlation coefficients and the corresponding p-values. As shown in Tables 1 and 2, all the coefficients are negative. The negative correlation is statistically significant, as demonstrated by many of the extremely small p-values.

[3]Throughout Section 3, we use SVD as the training model. For the hyper-parameters, the number of epochs is 50; the regularization weights of the latent factors λ_p and λ_q are both set to 0.02; the number of latent factors is 15.



(a) The Epinions dataset (b) The MovieLens-100K dataset

Fig. 1. The number of the received ratings of the items (in the training dataset) vs. the average RMSE scores (in the test dataset). Each dot represents the average RMSE score of the items that received the same number of ratings.



(a) The Epinions dataset (b) The MovieLens-100K dataset

Fig. 2. The number of the rated items of the users (in the training dataset) vs. the average RMSE scores (in the test dataset). Each dot represents the average RMSE score of the users who rated the same number of items.

To show the compound effect of both the number of ratings received of the items and the number of rated items of users on the RMSE scores, we show the heatmap of the two factors where the colors represent the magnitude of the averaged RMSE scores. For the Epinions dataset, the most active user rated 821 items in the training data, and the most rated item received 1,609 ratings in the training data. Since plotting the entire 1,609-by-821 grid makes the figure difficult to read, we divide each dimension into 20 equal-sized regions. Eventually, the heatmap is composed of

Table 1. The Pearson correlation coefficient (PCC) and the corresponding p-value between (1) x: the number of received ratings of an item (in the training data) and (2) y: the average RMSE score (in the test data) of the items that received x ratings

Dataset	PCC	p-value
Epinions	-0.45	1.5×10^{-16} (***)
MovieLens-100K	-0.70	1.8×10^{-37} (***)
FilmTrust	-0.73	2.0×10^{-14} (***)
Yahoo! Movies	-0.12	0.04 (*)
AMI	-0.36	3.6×10^{-10} (***)

Note: We show the results on five different public datasets.

Table 2. The Pearson's correlation coefficient (PCC) and the corresponding p-value between (1) x: the number of a user's rated items (in the training data) and (2) y: the average RMSE score (in the test data) of the users who rated x items

Dataset	PCC	p-value
Epinions	-0.51	3.5×10^{-23} (***)
MovieLens-100K	-0.52	9.4×10^{-19} (***)
FilmTrust	-0.28	0.012 (*)
Yahoo! Movies	-0.25	0.00026 (***)
AMI	-0.50	3.4×10^{-6} (***)

Note: We show the results on five different public datasets.

the 20-by-20 cells, and each cell represents the averaged RMSE scores of all the predicted ratings in this region. As shown in Figure 3(a) and (b), the RMSE score decreases (i.e., better prediction) as the user becomes more active or the items becomes more popular in both the Epinions and the MovieLens-100K datasets. Experimental results on other datasets show similar results so we skipped these figures to save space.

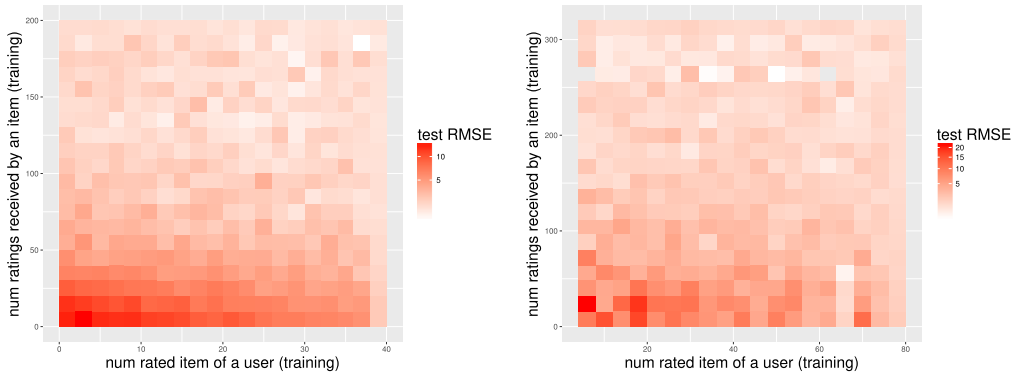
4 DIFFERENTIATING REGULARIZATION FUNCTIONS

This section presents our proposed method – differentiating the regularization weights for different items and users based on the differentiating regularization functions. We show the integration of this technique with the conventional SVD model in details. However, the proposed method can be easily integrated with other learning-based recommendation modules. As will be shown later in Section 5, we integrated this technique with several MF-based methods, including the SVD, SVD++, and NMF models, and compared their corresponding RMSE scores.

We list the symbols that will be used later and their definitions in Table 3.

4.1 Preliminary

4.1.1 Mean Model and Bias Model. To predict a user i's rating on an item j based on other known ratings, the simplest approach is probably the mean model, which always predicts the unknown ratings as the average of all the known ratings. Thus, the predicted rating $\hat{r}_{ij} = \mu = \sum_{\forall (i,j) \in \mathcal{K}} r_{ij} / |\mathcal{K}|$, where $|\mathcal{K}|$ returns the size of the set $\mathcal{K}$.



(a) The Epinions dataset (b) The MovieLens-100K dataset

Fig. 3. The heatmap of the RMSE scores given the numbers of users' rated items and the numbers of ratings received by the items. We use a 20×20 grid to divide the space. Each cell shows the RMSE score of all the ratings within the range.

Table 3. A List of Important Symbols and Their Definitions

Symbol	Definition
m	The number of users (given)
n	The number of items (given)
k	The number of latent factors (given)
$\boldsymbol{R} = [r_{ij}]$	The rating matrix with m rows and n columns $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$ (partially given)
$\hat{\boldsymbol{R}} = [\hat{r}_{ij}]$	The predicted rating matrix with m rows and n columns $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$ (to compute)
μ	The average of all known ratings (to compute)
$\boldsymbol{P} = [\boldsymbol{p}_i]^T$	The low-ranked matrix with m rows and k columns representing the latent factors of the users $i = 1, 2, \ldots, m$ (to compute)
$\boldsymbol{Q} = [\boldsymbol{q}_j]^T$	The low-ranked matrix with n rows and k columns representing the latent factors of the item $j = 1, 2, \ldots, n$ (to compute)
$b_i^{(U)}$	User i's user bias $i = 1, 2, \ldots, m$ (to compute)
$b_j^{(I)}$	Item j's item bias $j = 1, 2, \ldots, n$ (to compute)
$\mathcal{K}$	The set of all known (user, item) index pairs (given)
$\mathcal{K}_{test}$	The set of all known (user, item) index pairs assigned to the test dataset (given)
$\mathcal{K}_U$	The set of all known users (given)
$\mathcal{K}_I$	The set of all known items (given)
$R^{(U)}(i)$	The set of the items rated by user i (given)
$R^{(I)}(j)$	The set of the users who rated item j (given)
λ_x	The regularization term for the parameter x (given)

To make the model more realistic, one may consider to include (1) user bias $b_i^{(U)}$, which indicates user i's tendency to over-rate or under-rate the items compared to the other users, and (2) item bias $b_j^{(I)}$, which suggests an item j's tendency to receive a higher or lower score compared to the average score. Equation (1) shows the formula to predict an unknown rating.

$$\hat{r}_{ij} = \mu + b_i^{(U)} + b_j^{(I)}, \tag{1}$$

where $b_i^{(U)} = (\sum_{\forall j \in R^{(U)}(i)} r_{ij}/|R^{(U)}(i)|) - \mu$, and $b_j^{(I)} = (\sum_{\forall i \in R^{(I)}(j)} r_{ij}/|R^{(I)}(j)|) - \mu$.

4.1.2 SVD Model with Bias. The most typical MF model is sometimes called the SVD model in the literature,[4] although it is different from the classic singular value decomposition in linear algebra. The MF model compiles a rating matrix $\boldsymbol{R} = [r_{ij}]$ to record all the known ratings and attempts to find two low ranked matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ such that $\boldsymbol{P} \cdot \boldsymbol{Q}^T \approx \boldsymbol{R}$ based on the known entries in $\boldsymbol{R}$.

It is reported that combining the bias model and the SVD model yields better result. Therefore, a user i's rating on an item j is computed based on the following:

$$\hat{r}_{ij} = \mu + b_i^{(U)} + b_j^{(I)} + \boldsymbol{q}_j^T \cdot \boldsymbol{p}_i. \tag{2}$$

The two vectors $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$ represent the latent factors for the user i and the latent factors for the item j, respectively. If the two vectors are very similar, the inner-product operation would produce a large number, i.e., user i may assign a high rating to the item j. Likewise, if the latent factors of a user and an item are very different, Equation (2) tends to output a low rating.

Practically, only a very small portion of $\boldsymbol{R}$ contains values. The entries with no values should not simply be regarded as 0, because they are the ratings to be estimated. Since the matrix has many missing values, classic matrix decomposition techniques, such as singular value decomposition, LU decomposition, and QR decomposition, cannot directly be applied here. As a result, this is often modeled as a numerical optimization problem – finding the parameters to minimize a weighted sum of (1) the distances between the estimated rating and known ratings, and (2) the Frobenius norms of the parameters. Specifically, the objective function is shown as follows:

$$\frac{1}{2} \sum_{\forall (i,j) \in \mathcal{K}} (r_{ij} - \hat{r}_{ij})^2 + \frac{\lambda_p}{2} \sum_{\forall i \in \mathcal{K}_U} \|\boldsymbol{p}_i\|_2^2 + \frac{\lambda_q}{2} \sum_{\forall j \in \mathcal{K}_I} \|\boldsymbol{q}_j\|_2^2 + \frac{\lambda_U}{2} \sum_{\forall i \in \mathcal{K}_U} \left\|b_i^{(U)}\right\|_2^2 + \frac{\lambda_I}{2} \sum_{\forall j \in \mathcal{K}_I} \left\|b_j^{(I)}\right\|_2^2, \tag{3}$$

where $\hat{r}_{ij}$ is computed by Equation (2).

Popular optimization strategies include the SGD, the ALS, or their variations.

4.2 Differentiating Regularization Weights

As shown in Section 3, the SVD model better predicts the ratings to the popular items and the active users' ratings, probably because these items and users reveal more information. Therefore, we probably should assign lower constraints to the latent factors of these items and users. Likewise, we should assign higher constraints on the long tail items and the less active users to prevent their latent factors from being influenced by the few extreme observations.

The objective function of the conventional SVD method (as shown in Equation (3)), however, assigns a universal regularization weight λ_p to all the users' latent factors and another universal regularization weight λ_q to all the items' latent factors. This violates our argument above.

[4]The naming convention is probably from Simon Funk's blog post at http://sifter.org/simon/journal/20061211.html. This blog post popularizes the SVD model introduced here.

To make the model more consistent with our claim, we may extend the SVD model such that the values of the regularization weights inversely correlated with the number of ratings received by the items (and, likewise, inversely correlated with the number of users' rated items). As a result, the new objective function can be shown as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{\forall(i,j)\in\mathcal{K}}(r_{ij}-\hat{r}_{ij})^2 + \frac{\lambda_p}{2}\sum_{\forall i\in\mathcal{K}_U}\frac{1}{f(R^{(U)}(i))}\|\boldsymbol{p}_i\|_2^2 + \frac{\lambda_q}{2}\sum_{\forall j\in\mathcal{K}_I}\frac{1}{f(R^{(I)}(j))}\|\boldsymbol{q}_j\|_2^2 + \frac{\lambda_U}{2}\sum_{\forall i\in\mathcal{K}_U}\frac{1}{f(R^{(U)}(i))}\left\|b_i^{(U)}\right\|_2^2 + \frac{\lambda_I}{2}\sum_{\forall j\in\mathcal{K}_I}\frac{1}{f(R^{(I)}(j))}\left\|b_j^{(I)}\right\|_2^2, \tag{4}$$

where $R^{(U)}(i)$ is the set of items rated by the user i, $R^{(I)}(j)$ denotes the set of users who rated the item j, and $\boldsymbol{\theta} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m, \boldsymbol{q}_1, \ldots, \boldsymbol{q}_n, b_1^{(U)}, \ldots, b_m^{(U)}, b_1^{(I)}, \ldots, b_n^{(I)}]$, i.e., $\boldsymbol{\theta}$ represents all the parameters to learn.

In this article, the function $f(x)$ is called the RDF, because this function differentiates the regularization weights. If we set $f(x) = 1\ \forall x$, the above objective function is identical to Equation (3). However, to follow our claim, $f(x)$ should be a positive (i.e., $f(x) > 0$) and monotonically increasing (i.e., $\forall x \le y, f(x) \le f(y)$) function.

A linear function $f(x) = ax + b$ with a positive slope (i.e., $a > 0$) is probably the most straightforward choice of an increasing function. To ensure that $f(x)$ is always positive, one may consider to set the intercept to be non-negative (i.e., $b \ge 0$). When setting $b = 0$, the value of the regularization weight is proportional to the inverse of the number of a user's rated items and inverse to the number of the ratings received by an item.

Another possible choice of the RDF is the logarithm function, i.e., $f(x) = \log x$. This choice is motivated by the information theory: to encode k distinct entities, we would need $\log k$ bits. Additionally, as a user rated more items, the latest rated items are less valuable, compared to the first few rated items. This is mainly because a user can probably be characterized by the early rated items, and the extra information revealed by the newly rated items is only marginal. Likewise, the early ratings received by an item probably mostly reveal this item's information. As a result, we may want a sub-linear increasing function, such as the logarithm function, as the RDF. To prevent the function outputs zero or negative values, we may add an integer whose value no less than 1 to the argument, i.e., $f(x) = \log(x + c)(c \ge 1)$. The constant c may also serve as a smoothing factor.

Yet another possible choice of the RDF is the square root function, i.e., $f(x) = \sqrt{x}$. This choice is motivated by the SVD++ predicting function, in which the sum of the latent factors from the implicit feedback for a user i is normalized by the square root of the number of ratings from the user i [28]. Similar to the logarithm function, one may consider to add a positive constant c as the smoothing factor, i.e., $f(x) = \sqrt{x + c}$. The square root function grows even slower than the logarithm function. So, if it is believed that the new ratings reveal very few (extra) information about the users and the items, square root function could be a better choice than the logarithm function.

In summary, we propose the following three RDF for the following experiments.

$$f(x) = \begin{cases} ax + b\ (a > 0, b \ge 0), & \text{if use linear function} \\ \log(x + c)\ (c \ge \exp(1)), & \text{if use logarithm} \\ \sqrt{x + c}\ (c \ge 0), & \text{if use logarithm} \end{cases} \tag{5}$$

ALGORITHM 1: Training the SVD model with RDF based on SGD

Data: The rated (user, item) pairs $\mathcal{K}$, the regularization coefficients $\boldsymbol{\lambda} = (\lambda_p, \lambda_q, \lambda_U, \lambda_I)$, the learning rates $\boldsymbol{\eta} = (\eta_p, \eta_q, \eta_U, \eta_I)$, the learning decay rate γ

Result: Model parameters $\Theta = (\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{b}^{(U)}, \boldsymbol{b}^{(I)})$

$\boldsymbol{P} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{1})$; $\boldsymbol{Q} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{1})$; $\boldsymbol{b}^{(U)} \leftarrow \mathbf{0}$; $\boldsymbol{b}^{(I)} \leftarrow \mathbf{0}$; $epoch \leftarrow 0$;

repeat

  for $(i, j) \in \mathcal{K}$ **do**

    $\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i - \gamma^{epoch} \eta_p \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{p}_i}$ (based on Equation (6));

    $\boldsymbol{q}_j \leftarrow \boldsymbol{q}_j - \gamma^{epoch} \eta_q \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{q}_j}$ (based on Equation (7));

    $b_i^{(U)} \leftarrow b_i^{(U)} - \gamma^{epoch} \eta_U \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial b_i^{(U)}}$ (based on Equation (8));

    $b_j^{(I)} \leftarrow b_j^{(I)} - \gamma^{epoch} \eta_I \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial b_j^{(I)}}$ (based on Equation (9));

  end

  $epoch \leftarrow epoch + 1$;

until *termination condition is met*;

4.3 Training Algorithm

To obtain the parameters (i.e., each user's latent factors $\boldsymbol{p}_i$ and user bias $b_i^{(U)}$; each item's latent factors $\boldsymbol{q}_j$ and item bias $b_j^{(I)}$) to minimize the objective function $\mathcal{L}(\boldsymbol{\theta})$ (as shown in Equation (4)), we apply the SGD approach. Therefore, we would need to compute the partial derivative of $\mathcal{L}(\boldsymbol{\theta})$ to all the parameters, as shown below, and update the parameters toward the inverse directions of the corresponding gradients:

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{p}_i} = (r_{ij} - \hat{r}_{ij})\boldsymbol{q}_j + \frac{\lambda_p}{f(R^{(U)}(i))}\boldsymbol{p}_i \tag{6}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{q}_j} = (r_{ij} - \hat{r}_{ij})\boldsymbol{p}_i + \frac{\lambda_q}{f(R^{(I)}(j))}\boldsymbol{q}_j \tag{7}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial b_i^{(U)}} = (r_{ij} - \hat{r}_{ij}) + \frac{\lambda_U}{f(R^{(U)}(i))}b_i^{(U)} \tag{8}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial b_j^{(I)}} = (r_{ij} - \hat{r}_{ij}) + \frac{\lambda_I}{f(R^{(I)}(j))}b_j^{(I)} \tag{9}$$

The entire training process is shown in Algorithm 1.

Since the proposed technique – differentiating regularization function – is simple and general, it can be employed on many rating prediction models, such as the SVD++ model, the NMF model, the Prod2Vec model, the Behavior2Vec model, and so on.

4.4 Model Complexity and Time Complexity

If we apply the RDF on the SVD model, as shown by the Algorithm 1, the parameters to learn includes $\boldsymbol{P} \in \mathcal{R}^{m \times k}$, $\boldsymbol{Q} \in \mathcal{R}^{n \times k}$, $\boldsymbol{b}^{(U)} \in \mathcal{R}^{m \times 1}$, and $\boldsymbol{b}^{(I)} \in \mathcal{R}^{n \times 1}$. Therefore, the number of parameters to learn is $mk + nk + m + n = (m + n)(k + 1)$, the same as in the SVD model.

For the time complexity of training, every update of $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$, as shown by Algorithm 1, Equation (6), and Equation (7) requires $O(k)$, and every update of $b_i^{(U)}$ and $b_j^{(I)}$ costs $O(1)$. Therefore, the time complexity of each epoch is $O(|\mathcal{K}|\, k)$. Assuming we need T epochs to meet the

Table 4. Statistics of the Benchmark Datasets

Dataset	# users	# items	# ratings	Density	Rating scale
Epinions	40,163	139,738	664,824	0.0118%	[1, 2, 3, 4, 5]
MovieLens-100K	943	1,682	100,000	6.3047%	[1, 2, 3, 4, 5]
FilmTrust	1,508	2,071	35,497	1.1366%	[0.5, 1, 1.5, . . . , 4]
Yahoo! Movies	7,642	11,916	221,367	0.2431%	[1, 2, . . . , 13]
AMI	339,231	83,046	500,176	0.0018%	[1, 2, 3, 4, 5]

termination condition, the total training time complexity would be $O(T\,|\mathcal{K}|\,k)$, which is the same as the time complexity of the training algorithm of SVD.

To predict a user i's rating on an item j, the predicting function is shown by Equation (2), which requires to compute the inner product of two vectors $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$. Therefore, the time complexity is $O(k)$, which is very efficient, since a typical k is not large.

For similar reasons, if we apply the RDF on other MF-based methods (e.g., SVD++ or NMF, as we will shown in Section 5), their corresponding model complexity, training time, and prediction time will be the same as the original methods.

The model complexity of SVD++ is much larger than the SVD model, because the SVD++ model requires to obtain the latent factors of the implicit feedbacks as the extra parameters. As we will show in Section 5, although SVD with RDF is simpler (in terms of model complexity), the predictions are more accurate compared to the conventional SVD++ model (i.e., without incorporating with RDF). This makes incorporating the RDF to simple models an attractive choice in practice, because such a combination is fast in both training and predicting and still maintain a comparable (in many cases, even better) accuracy to the complex models.

5 EXPERIMENTS

5.1 Description of the Benchmark Datasets

We collected public datasets from various domains as the benchmark datasets, including the following: (1) the Epinions dataset[5] – a rich product review dataset on the Epinions website [38]; (2) the MovieLens-100K dataset[6] – a movie rating dataset collected and released by the GroupLens group [21]; (3) the FilmTrust dataset[7] – a dataset crawled from the entire FilmTrust website [19]; (4) Yahoo! Movies dataset[8] – a sample of users' ratings on the movies released by Yahoo! Movies [36, 37]; (5) the AMI dataset[9] – an Amazon review information on the musical instruments [22]. The statistical summaries of these datasets are shown in Table 4.

For each of the collected dataset, we randomly assign 80% of the ratings as the training data and the remaining 20% as the test data. In all the following experiments, the reported metrics (e.g., the RMSE scores) are based on the results on the test data.

5.2 Experiment Settings

We compared the SVD model, SVD++ model, and the NMF model with and without the RDFs. For each of the compared methods, we performed a grid search on the set of the hyper-parameters. Particularly, we divided the training data into two sets – the training set and the validation set.

[5] http://www.epinions.com/.
[6] https://grouplens.org/.
[7] https://www.librec.net/datasets.html.
[8] https://webscope.sandbox.yahoo.com/catalog.php?datatype=r.
[9] http://jmcauley.ucsd.edu/data/amazon/.

Table 5. RMSE Scores of SVD and SVD with Regularization Weights on the Test Datasets

Dataset	SVD	Linear-reg	sqrt-reg	log-reg	Improve ratio range
Epinions	1.1997	**1.0538** (***)	**1.0538** (***)	**1.0538** (***)	12.16%
MovieLens-100K	0.9423	**0.9422**	**0.9422**	**0.9422**	0.01%
FilmTrust	0.8465	**0.8194** (***)	**0.8194** (***)	0.8223 (***)	2.86% to 3.20%
Yahoo! Movies	3.0799	**2.9892** (***)	3.0129 (***)	3.0127 (***)	2.18% to 2.94%
AMI	1.1450	**1.1405** (***)	**1.1405** (***)	**1.1405** (***)	0.39%

Most results are significantly better than the SVD model ($p < 0.001$).

We trained the models with combinations of the hyper-parameter candidates on the training set and computed the RMSE scores based on the validation set. We recorded the hyper-parameter combination that produces the best result (i.e., the lowest RMSE score) on the validation set and used such hyper-parameters to re-train the model based on the entire training data (i.e., the training set plus the validation set), and reported the performance of the model based on the test data.

Specifically, for the conventional SVD model and the SVD model with the RDFs, we set the candidate regularization weights for the user bias and the item bias (i.e., λ_U and λ_I in Equations (3) and (4)) as the following values: [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000], and we set the candidate regularization weights for the latent factors (i.e., λ_p and λ_q in Equations (3) and (4)) as the following values: [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000]. As a result, we tested 121 different hyper-parameter combinations. For the SVD++ model and the SVD++ model with the RDF, we set the regularization weights for the user bias and item bias to 0.1; we set the candidate regularization weights for the latent factors to the following values: [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000]; we set the candidate regularization weights for the latent factors of the implicit feedback as the following values: [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000]. As a result, we tested 121 different hyper-parameter settings. For the NMF model and the NMF model with the RDF, we set λ_U and λ_I as the following values: [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000], and we set the candidate regularization weights for the latent factors as the following values: [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000]. As a result, we tested 121 different hyper-parameter combinations. For all the models, we set k the numbers of latent factors to 15, γ the learning decay rate in the Algorithm 1 to 0.9, and the number of training epochs to 50. However, for most of the experiments, the parameters started to converge in less than 20 epochs.

5.3 The Overall Predictability

We quantify $S(M)$ the predicting score of a method M based on the RMSE score between M's predicted ratings (i.e., $\hat{\boldsymbol{R}}^{(M)}$) and the actual ratings (i.e., $\boldsymbol{R}$), as defined by the following:

$$\begin{aligned} S(M) &= \text{RMSE}\left(\boldsymbol{R}, \hat{\boldsymbol{R}}^{(M)}\right) \\ &= \sqrt{\frac{1}{|\mathcal{K}_{test}|} \sum_{\forall (i,j) \in \mathcal{K}_{test}} \left(r_{ij} - \hat{r}_{ij}^{(M)}\right)^2}. \end{aligned} \tag{10}$$

We further defined the improve ratio from baseline method M_{base} to the new method M_{new} as follows:

$$IR(M_{new}, M_{base}) = \frac{S(M_{base}) - S(M_{new})}{S(M_{base})}. \tag{11}$$

Table 5 shows the comparison of the conventional SVD model with the SVD model integrated with three RDFs – linear (linear-reg), square root (sqrt-reg), and logarithm (log-reg). As can be seen, integrating the RDFs into the SVD model makes the predictions more accurate (i.e., lower RMSE

Table 6. RMSE Scores of SVD++ and SVD++ with Regularization Weights on the Test Datasets

Dataset	SVD++	Linear-reg	sqrt-reg	log-reg	Improve ratio range
Epinions	1.0628	**1.0538** (***)	1.0566 (***)	**1.0538**	0.58% to 0.85%
MovieLens-100K	0.9423	**0.9422**	**0.9422**	0.9423	0.00% to 0.01%
FilmTrust	0.8199	0.8199	0.8197	**0.8194**	0.00% to 0.06%
Yahoo! Movies	3.0152	3.0127 (***)	**3.0126** (***)	3.0128 (***)	0.08% to 0.09%
AMI	1.1446	**1.1407** (***)	1.1429 (***)	1.1408 (***)	0.15% to 0.34%

Even compared to the competative baseline SVD++, many results are still significantly better ($p < 0.001$).

Table 7. RMSE Scores of NMF and NMF with Regularization Weights on the Test Datasets

Dataset	NMF	Linear-reg	sqrt-reg	log-reg	Improve ratio range
Epinions	1.0709	**1.0582** (***)	**1.0582** (***)	1.0583 (***)	1.18% to 1.19%
MovieLens-100K	0.9621	0.9416 (***)	0.9416 (***)	**0.9413** (***)	2.13% to 2.16%
FilmTrust	0.9295	**0.8206** (***)	**0.8206** (***)	**0.8206** (***)	11.72%
Yahoo! Movies	3.6058	**3.0140** (***)	3.0141 (***)	3.0141 (***)	16.41%
AMI	**1.1431**	1.1433	1.1432	1.1432	−0.01% to −0.02%

Most results are significantly better than the NMF model ($p < 0.001$).

scores) in all cases. We highlight the best (i.e., the lowest RMSE scores) results for each benchmark dataset. We conducted similar comparisons on the SVD++ model (with and without the RDFs) and the NMF model (with and without the RDFs). The results are reported in Tables 6 and 7. Again, the models integrated with the RDFs outperforms the baselines in almost all cases. This suggests that indeed differentiating the regularization weights in the MF models for the recommender systems is beneficial.

We observed that, if the RDFs are not introduced, the simple models (i.e., the SVD model and the NMF model in this article) usually perform worse than the complex models (i.e., the SVD++ model in this article). However, when the simple models are integrated with the RDFs, they outperform the complex models that do not integrated with the RDFs in many cases. As shown by the last columns of Tables 5 and 6, the improve ratios of the SVD model and the NMF model are evident. In the meanwhile, the improvement of applying the RDFs on the complex models (i.e., the SVD++ model in this article) is less obvious. Since we tested hyper-parameters for all the models based on grid search, as reported in Section 5.2, even in the cases where the improve ratios are small, these improvements are consistent and stable.

Since the complex models require longer training and predicting time, training simpler models with the RDFs should be an attractive choice, especially for the large-scale recommender systems that typically have millions of users and items, and tens to hundreds of millions of interactions among users and items.

In addition to the RMSE scores, we also show a comparison of the other metrics for the Epinions dataset in Table 8. Specifically, we show the Mean Absolute Error (MAE) scores and the R^2 scores of the SVD model with and without the RDFs. The definitions of these two metrics are shown in Equations 12 and 13 as follows:

$$\text{MAE}\left(\boldsymbol{R}, \hat{\boldsymbol{R}}^{(M)}\right) = \frac{1}{|\mathcal{K}_{test}|} \sum_{\forall (i,j) \in \mathcal{K}_{test}} \left| r_{ij} - \hat{r}_{ij}^{(M)} \right|, \tag{12}$$

$$R^2\left(\boldsymbol{R}, \hat{\boldsymbol{R}}^{(M)}\right) = 1 - \frac{\sum_{\forall (i,j) \in \mathcal{K}_{test}} \left(r_{ij} - \hat{r}_{ij}^{(M)} \right)^2}{\sum_{\forall (i,j) \in \mathcal{K}_{test}} (r_{ij} - \mu)^2}. \tag{13}$$

Table 8. Various Metrics of SVD and SVD with Regularization Weights on the Epinions Dataset

Metrics	SVD	Linear-reg	sqrt-reg	log-reg	Improve ratio range
RMSE	1.1997	**1.0538** (***)	**1.0538** (***)	**1.0538** (***)	12.16%
MAE	0.9139	**0.8190** (***)	**0.8190** (***)	**0.8190** (***)	10.38%
R^2	0.0067	**0.2337** (***)	**0.2337** (***)	**0.2337** (***)	3388.06%

All the improvements are significant.

Table 9. RMSE Scores (in the Test Data) of SVD and SVD with Regularization Weights on the 10% of Items that Received the Fewest Number of Ratings (in the Training Data)

Dataset	SVD	Linear-reg	sqrt-reg	log-reg	Improve ratio range
Epinions	2.2505 (***)	2.1189 (***)	**2.1186** (***)	2.1187 (***)	5.85% to 5.86%
MovieLens-100K	1.8866	1.8781 (**)	**1.8785**	1.8808	0.31% to 0.45%
FilmTrust	1.8454	1.7246	**1.7204**	1.8122	1.80% to 6.77%
Yahoo! Movies	26.1175	**25.4348** (**)	25.5993	25.5766 (*)	1.98% to 2.61%
AMI	3.0426	3.0436	**3.0406**	3.0448	−0.07% to 0.07%

Table 10. RMSE Scores (in the Test Data) of SVD++ and SVD++ with Regularization Weights on the 10% of Items that Received the Fewest Number of Ratings (in the Training Data)

Dataset	SVD++	Linear-reg	sqrt-reg	log-reg	Improve ratio range
Epinions	2.2295	2.1373	**2.1179** (***)	2.1186 (**)	4.14% to 5.01%
MovieLens-100K	1.8809	1.8797	**1.8757**	1.8784	0.06% to 0.28%
FilmTrust	1.7523	1.7296	1.7337	**1.7180**	1.06% to 1.96%
Yahoo! Movies	26.2629	25.5760 (*)	**25.5558** (*)	25.5704 (*)	2.62% to 2.69%
AMI	3.1112	**3.0448** (***)	3.0480 (***)	3.0456 (***)	2.03% to 2.13%

Both the RMSE and the MAE are negative-oriented scores, i.e., lower values are better. However, the R^2 score is positive-oriented, i.e., higher values are better. Therefore, the improve ratio of the R^2 score is somewhat different – a negative sign should be added to Equation (11) when computing the improve ratio for the R^2 score.

To save the space, we only show the MAE scores and the R^2 scores of the SVD model with and without RDFs on the Epinions dataset. For the other models and the other datasets, the improve ratio of the MAE is usually larger than the RMSE, and the improve ratio of the R^2 score is usually the most manifest.

5.4 Predictability on the Long Tail Items and the Less Active Users

This section analyzes the effect of the RDFs on the long tail items and the less active users. Specifically, we show the RMSE scores of these items/users before and after applying the RDF on the SVD, SVD++, and NMF models. We expect that, in the cases where an item received few ratings or a user rated few items, applying the RDFs helps the predictions. As a result, applying the RDFs may partially solve the cold-start problem, which is a prevalent problem in the recommender systems.

For each of the experimental dataset, we extracted 10% of the items that received the fewest numbers of ratings in the training data. We computed the RMSE scores of the ratings of these items in the test data based on the SVD model with and without the RDFs. The results are shown in Table 9. As reported, the SVD models with the RDFs outperform the conventional SVD model in most cases. We also applied RDFs on the SVD++ and the NMF models, as reported in Tables 10

Table 11. RMSE Scores (in the Test Data) of NMF and NMF with Regularization Weights on the 10% of Items that Received the Fewest Number of Ratings (in the Training Data)

Dataset	NMF	log-reg	Linear-reg	sqrt-reg	Improve ratio range
Epinions	2.0793	2.0757	2.0753	**2.0750** (*)	0.17% to 0.21%
MovieLens-100K	2.0788	**1.8683** (***)	1.8749 (***)	1.8724 (***)	9.81% to 10.13%
FilmTrust	1.8428	**1.6952** (*)	1.7026	1.6956 (*)	7.61% to 8.01%
Yahoo! Movies	32.8463	**25.4302** (***)	25.4408 (***)	25.4314 (***)	22.55% to 22.58%
AMI	2.9942	**2.9939**	2.9945	2.9950	−0.03% to 0.01%

Table 12. RMSE Scores (in the Test Data) of SVD and SVD with Regularization Weights on the 10% of Users Whose Number of Ratings Are the Fewest (in the Training Data)

Dataset	SVD	Linear-reg	sqrt-reg	log-reg	Improve ratio range
Epinions	2.7654	2.2856 (***)	**2.2844** (***)	2.2850 (***)	17.35% to 17.39%
MovieLens-100K	**1.6230**	1.6249	1.6261	1.6256	−0.12% to −0.19%
FilmTrust	1.5136	**1.2352** (**)	1.2487 (**)	1.5971	−5.52% to 18.39%
Yahoo! Movies	**20.1134**	20.1687	20.2836	20.2740	−0.27% to −0.85%
AMI	2.5518	2.5454	**2.5441**	2.5445	0.25% to 0.30%

Table 13. RMSE Scores (in the Test Data) of SVD++ and SVD++ with Regularization Weights on the 10% of Users Whose Number of Ratings Are the Fewest (in the Training Data)

Dataset	SVD++	Linear-reg	sqrt-reg	log-reg	Improve ratio range
Epinions	**2.2847**	2.3885 (**)	2.3244	2.2877	−0.13% to −4.54%
MovieLens-100K	**1.6219**	1.6245	1.6259	1.6235	−0.10% to −0.25%
FilmTrust	1.2715	1.2512	1.2464	**1.2244**	1.60% to 3.70%
Yahoo! Movies	20.2961	20.2808	20.2819	**20.2726** (*)	0.07% to 0.12%
AMI	2.6267	**2.5465** (*)	2.5975	2.5524 (*)	1.11% to 3.05%

and 11. As before, applying the RDFs appears to better predict the long tail items in most cases. However, compared to the overall RMSE scores reported in Tables 5, 6, and 7, the predictions on the long tail items is still less accurate than an average item.

Likewise, for each of the experimental dataset, we extracted 10% of the users who rated fewest items in the training data. We computed the RMSE scores of the ratings from these users in the test data. In Table 12, we report the result of the SVD model with and without the RDFs. It appears that the effect of using the number of users' rated items as the input of the RDFs is unstable – such a mechanism may help the predictions for certain datasets (e.g., on the Epinions dataset and the AMI dataset) but sometimes may make things worse (e.g., on the MovieLens-100K dataset and the Yahoo! Movies dataset). We also compared the SVD++ model and the NMF model with and without RDFs (using the number of users' rated items as the inputs for the RDFs). The results are reported in Tables 13 and 14. It seems that using the number of a user's rated items as the input of the RDFs may not necessarily improve the recommendation quality.

To further compare the performance of the SVD models with and without the differentiating functions with respect to (1) the number of ratings received by each item and (2) each user's number of ratings, we plotted the heatmap to visualize the result. Figure 4 shows the result based on the Epinions dataset. Specifically, the most rated item received 1,609 ratings in the training data, and the most active user in this dataset rated 821 items in the training data . Like what we did in

Table 14. RMSE Scores (in the Test Data) of NMF and NMF with Regularization Weights on the 10% of Users Whose Number of Ratings Are the Fewest (in the Training Data)

Dataset	NMF	Linear-reg	sqrt-reg	log-reg	Improve ratio range
Epinions	2.2689	2.2575	**2.2566**	2.2585	0.46% to 0.54%
MovieLens-100K	**1.5661**	1.5998	1.6005	1.6006	−2.15% to −2.20%
FilmTrust	1.3275	1.2440	**1.2402**	1.2430	6.29% to 6.58%
Yahoo! Movies	24.7288	20.2169 (***)	**20.2160** (***)	20.2193 (***)	18.24% to 18.25%
AMI	**2.4633**	2.4638	2.4644	2.4634	−0.04% to 0.00%

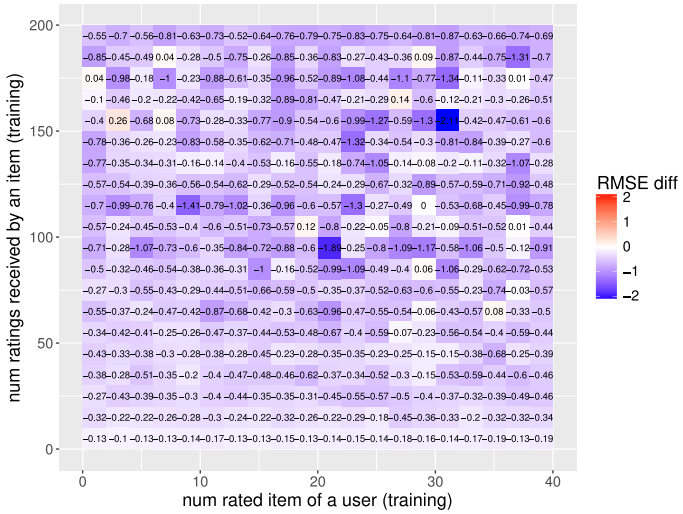


Fig. 4. The difference between the RMSE score of the SVD model with and without the logarithm differentiating function (based on the Epinions dataset). Negative values (i.e., the blue cells) indicate that applying the logarithm differentiating function is better in that region.

Section 3, we divided each dimension into 20 equal-sized regions. For each test instance r_{ij} falls in the cell (u, v) on the heatmap, we compute $r_{ij}^{(\text{SVD})}$ the rating predicted by the SVD model and $r_{ij}^{(\text{df})}$ the rating predicted by including the RDF. We computed the RMSE score for all the ratings within a cell for all the cells based on the SVD models with and without the RDF. Finally, we subtract the two RMSE scores for each cell to get the performance difference. Equation (14) shows the formula:

$$s_{uv} = \sqrt{\frac{1}{|A(u,v)|} \sum_{\forall (i,j) \in A(u,v)} \left(r_{ij} - \hat{r}_{ij}^{(\text{df})}\right)^2} - \sqrt{\frac{1}{|A(u,v)|} \sum_{\forall (i,j) \in A(u,v)} \left(r_{ij} - \hat{r}_{ij}^{(\text{SVD})}\right)^2}, \quad (14)$$

where $A(u, v)$ returns the set of all the test instances (i.e., user i's rating on item j) in the uth row and vth column in the heatmap, and $|A(u, v)|$ denotes the size of the set.

Each cell in Figure 4 shows the difference between the two RMSE scores. Since a lower RMSE score means better prediction, the negative values (i.e., the blue color cells) means the weight differentiating mechanism has more accurate predictions compared to the original SVD model. As can be seen, the SVD model with RDF has more accurate predictions as we increase the value of y (the number of ratings received by each item). However, as we increase the value of x (the number of rated items of a user), we do not see obvious trend. Here, we only show the logarithm

differentiating function on the Epinions dataset. However, experiments on the other differentiating functions and the other benchmark datasets show similar results. The results demonstrate that by applying the differentiating functions, we can better predict the long tail items.

6 DISCUSSION AND FUTURE WORK

This article discusses our study on CF methods incorporating with the RDF such that different items and users may have different regularization weights on their corresponding latent factors. We proposed three RDF, which takes number of the ratings received by an item or the number of a user's rated items as the inputs, to decide the regularization weights of the latent factors of the item and the user. We compared the proposed methods with the conventional CF methods on five open datasets. The results show that the proposed method better predict users' ratings on the items. More importantly, incorporating the proposed method with various CF methods better predict the less informative items (i.e., the items received few ratings). As a result, the cold-start problem, which is prevalent in various recommender systems, could be partially alleviated.

Although we only compare the SVD, SVD++, and NMF models with and without RDFs, the technique proposed in this article can be applied on a wide range of learning-based recommendation models, e.g., FM, FFM, FPMC, and so on. The proposed RDFs can also be integrated with many previous approaches that aim to solve or partially solve the cold-start problem. In fact, as long as the objective function of a recommendation algorithm contains the regularization terms for the parameters of the users and/or the items, the RDFs can be applied. Therefore, the proposed method can be and should be a general technique to improve the predictions of the learning-based recommendation modules, but not a replacement to the existing recommendation models.

While it seems obvious that we should assign different regularization weights based on the size of the training dataset, the characteristic of the recommender systems is somewhat different – a larger training dataset may not necessarily reveal more information of an item or one individual user. As shown in [3], the interaction between users and items follow a long tail, i.e., few items received many ratings (likewise few users rated many items), but most items received few ratings (likewise most users rated few items). As a result, a large dataset may contain many information of the popular items and the highly active users, but few information on most items and users. Therefore, we probably should assign regularization weights item-wise and user-wise, as reported in this article. This is different from most machine learning models that adjust the regularization weights based on the number of available training instances and the model complexity.

While the RDF looks successful for the CF-based methods, we found no obvious winner among the three proposed functions in this study. We are interested in continuing investigating the properties of different RDF and their relationship with the distribution of the training data. We are also interested in theoretical studies on the RDF as a future work.

REFERENCES

[1] Fabian Abel, Eelco Herder, Geert-Jan Houben, Nicola Henze, and Daniel Krause. 2013. Cross-system user modeling and personalization on the social web. *User Modeling and User-Adapted Interaction* 23, 2–3 (2013), 169–209.

[2] Lukas Brozovsky and Vaclav Petricek. 2007. Recommender system for online dating service. arXiv: cs/0703042.

[3] Erik Brynjolfsson, Yu Hu, and Michael D. Smith. 2003. Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers. *Management Science* 49, 11 (2003), 1580–1596.

[4] Bin Cao, Nathan Nan Liu, and Qiang Yang. 2010. Transfer learning for collective link prediction in multiple heterogenous domains. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 159–166.

[5] Rich Caruana, Steve Lawrence, and C. Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems*. 402–408.

[6] Hung-Hsuan Chen. 2017. Weighted-SVD: Matrix factorization with weights on the latent factors. arXiv:1710.00482.

[7] Hung-Hsuan Chen. 2018. Behavior2Vec: Generating distributed representations of users behaviors on products for recommender systems. *ACM Transactions on Knowledge Discovery from Data* 12, 4 (2018), 43.

[8] Hung-Hsuan Chen, Chu-An Chung, Hsin-Chien Huang, and Wen Tsui. 2017. Common pitfalls in training and evaluating recommender systems. *ACM SIGKDD Explorations Newsletter* 19, 1 (2017), 37–45.
[9] Hung-Hsuan Chen and C. Lee Giles. 2015. ASCOS++: An asymmetric similarity measure for weighted networks to address the problem of simrank. *ACM Transactions on Knowledge Discovery from Data* 10, 2 (2015), 15.
[10] Hung-Hsuan Chen, Liang Gou, Xiaolong Zhang, and Clyde Lee Giles. 2011. Collabseer: A search engine for collaboration discovery. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*. ACM, 231–240.
[11] Hung-Hsuan Chen, II Ororbia, G. Alexander, and C. Lee Giles. 2015. ExpertSeer: A keyphrase based expert recommender for digital libraries. arXiv:1511.02058.
[12] Vladimir Cherkassky, Xuhui Shao, Filip M. Mulier, and Vladimir N. Vapnik. 1999. Model complexity control for regression using VC generalization bounds. *IEEE Transactions on Neural Networks* 10, 5 (1999), 1075–1089.
[13] Bradley Efron and Robert Tibshirani. 1986. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science* 1, 1 (1986), 54–75.
[14] Martin J. Eppler and Jeanne Mengis. 2004. The concept of information overload: A review of literature from organization science, accounting, marketing, MIS, and related disciplines. *Information Society* 20, 5 (2004), 325–344.
[15] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of Statistical Learning*. Vol. 1. Springer series in statistics New York.
[16] Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 69–77.
[17] Liang Gou, Jung-Hyun Kim, Hung-Hsuan Chen, Jason Collins, Marc Goodman, Xiaolong Luke Zhang, and C. Lee Giles. 2009. MobiSNA: A mobile video social network application. In *Proceedings of the 8th ACM International Workshop on Data Engineering for Wireless and Mobile Access*. ACM, 53–56.
[18] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1809–1818.
[19] G. Guo, J. Zhang, and N. Yorke-Smith. 2013. A novel Bayesian similarity measure for recommender systems. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. 2619–2625.
[20] Greg Hamerly and Charles Elkan. 2004. Learning the k in k-means. In *Advances in Neural Information Processing Systems*. 281–288.
[21] F. Maxwell Harper and Joseph A. Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2016), 19.
[22] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 507–517.
[23] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.
[24] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. arXiv:1511.06939.
[25] Arthur E. Hoerl and Robert W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12, 1 (1970), 55–67.
[26] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 43–50.
[27] Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Vol. 2. Montrea, 1137–1143.
[28] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* 42, 8 (2009), 30–37.
[29] Nick Landia. 2017. Building recommender systems for fashion: Industry talk abstract. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 343–343.
[30] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. 2012. Finding a needle in a haystack of reviews: Cold start context-based hotel recommender system. In *Proceedings of the 6th ACM Conference on Recommender Systems*. ACM, 115–122.
[31] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*. 2177–2185.
[32] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 661–670.

[33] Cheng-You Lien, Guo-Jhen Bai, Ting-Rui Chen, and Hung-Hsuan Chen. 2017. Predicting user's online shopping tendency during shopping holidays. (2017).
[34] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. 2014. Facing the cold start problem in recommender systems. *Expert Systems with Applications* 41, 4 (2014), 2065–2073.
[35] Jakub Macina, Ivan Srba, Joseph Jay Williams, and Maria Bielikova. 2017. Educational question routing in online student communities. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 47–55.
[36] Benjamin M. Marlin and Richard S. Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the Third ACM Conference on Recommender Systems*. ACM, 5–12.
[37] Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. 2007. Collaborative filtering and the missing at random assumption. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI'07)*. AUAI Press, Arlington, Virginia, 267–275.
[38] Paolo Massa, Kasper Souren, Martino Salvetti, and Danilo Tomasoni. 2001. Trustlet, open research on trust metrics. *Scalable Computing: Practice and Experience* 9, 4 (2001), 31–43.
[39] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.
[40] Aditya Krishna Menon and Charles Elkan. 2011. Link prediction via matrix factorization. In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 437–452.
[41] Samaneh Moghaddam and Martin Ester. 2013. The FLDA model for aspect-based opinion mining: Addressing the cold start problem. In *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 909–918.
[42] Andrew Y. Ng. 1998. On feature selection: Learning with exponentially many irrelevant features as training examples. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 404–412.
[43] Charles A. O'reilly. 1980. Individuals and information overload in organizations: Is more necessarily better?*Academy of Management Journal* 23, 4 (1980), 684–696.
[44] Seung-Taek Park and Wei Chu. 2009. Pairwise preference regression for cold-start recommendation. In *Proceedings of the 3rd ACM Conference on Recommender Systems*. ACM, 21–28.
[45] Michael P. Perrone and Leon N. Cooper. 1995. When networks disagree: Ensemble methods for hybrid neural networks. In *How We Learn; How We Remember: Toward An Understanding Of Brain And Neural Systems: Selected Papers of Leon N Cooper*. World Scientific, 342–358.
[46] Ioannis Psorakis, Stephen Roberts, Mark Ebden, and Ben Sheldon. 2011. Overlapping community detection using bayesian non-negative matrix factorization. *Physical Review E* 83, 6 (2011), 066114.
[47] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. 2002. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*. ACM, 127–134.
[48] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning preferences of new users in recommender systems: An information theoretic approach. *Acm Sigkdd Explorations Newsletter* 10, 2 (2008), 90–100.
[49] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 10th International Conference on Data Mining (ICDM '10)*. IEEE, 995–1000.
[50] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 811–820.
[51] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 253–260.
[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[53] Mingxuan Sun, Fuxin Li, Joonseok Lee, Ke Zhou, Guy Lebanon, and Hongyuan Zha. 2013. Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*. ACM, 445–454.
[54] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 58, 1 (1996), 267–288.
[55] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. ACM, 495–503.
[56] Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.

[57] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 315–324.

[58] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67, 2 (2005), 301–320.

Received June 2018; revised August 2018; accepted October 2018