Expert Systems With Applications 141 (2020) 112885

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# Robust weighted SVD-type latent factor models for rating prediction

Yiqi Gu[a], Xi Yang[a], Mengjiao Peng[b], Guang Lin[a,c,*]

[a] Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907, USA
[b] Department of Statistics, Purdue University, 250 N. University Street, West Lafayette, IN 47907, USA
[c] School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA

ARTICLE INFO

*Article history:*
Received 27 January 2019
Revised 16 August 2019
Accepted 17 August 2019
Available online 7 September 2019

*Keywords:*
Recommender system
Collaborative filtering
Latent factor model
Singular value decomposition
Weighting technique

ABSTRACT

Recommending system is a popular tool in many commercial or social platforms which finds interesting products for users based on their preference history. Predicting the ratings of items, such as movies, plays an essential role in the recommending system. In this context, we develop a new type of latent factor models by attaching weights to the entries of the incomplete ratings matrix. The weights are computed after estimating the user/item mean errors caused by the basic SVD model under the low-rank assumption on the ratings matrix. To accelerate the optimization process of our proposed models and other existing SVD-type models, a special design of the initial guess is suggested. In the experiments on real-world datasets, the proposed weighted models outperform other SVD-type methods, and the usage of the special initial guess improves the optimization significantly, obtaining lower MRSEs within fixed number of iterations, in comparison with the random initial guess. Furthermore, artificially noised datasets are taken to evaluate the methods, where the weighted models still perform better than other SVD-type models, implying their effectiveness and robustness in noised environment.



## 1. Introduction

Recommending system (RS) is widely utilized in E-commerce service, social media and entertainment applications to help to find the most interesting stuff for users (Breese, Heckerman, & Kadie, 1998; Leskovec, Rajaraman, & Ullman, 2014; Sarwar, Karypis, Konstan, & Riedl, 2000; Su & Khoshgoftaar, 2009; Walczak, 2003). For example, in movie websites, RS usually collects and analyzes the historical behaviors (e.g. rating a movie) of users and then predicts the unknown preference of them, by which the most interesting movies for each individual user can be recommended (Marovic, Mihokovic, Miksa, Pribil, & Tus, 2011). In many situations, the multi-class rating system is provided, allowing users to rate each item from score 1.0 (the least interesting) to 5.0 (the most interesting). Mostly, compared to the huge amount of total items, the number of rated items from a user is highly limited, causing the whole dataset sparse. The task of RS is to predict the rating score from each user to the items not rated so far by analyzing the whole data of existed ratings.

Collaborative filtering (CF) is a popular group of methods employed to build effective RS. Traditional CF techniques (Breese et al., 1998; Chee, Han, & Wang, 2001; Frmal & Lecron, 2017; Melville, Mooney, & Nagarajan, 2002; Sarwar, Karypis, Konstan, & Riedl, 2001; Su & Khoshgoftaar, 2006) are mostly memory-based, which perform the prediction by computing similarity between users or items and deploying weighted average. Such CF systems can be simply implemented and developed. However, they excessively rely on human ratings and perform worse if the dataset is highly sparse or of large scale. Another category of CF is model-based (Delporte, Karatzoglou, Matuszczyk, & Canu, 2013; Gao, Tang, Hu, & Liu, 2015; Ju & Xu, 2013; Koren, 2008; Salah, Rogovschi, & Nadif, 2016; Wang & Ke, 2014; Xu, Bu, Chen, & Cai, 2012). They predict the preference by building and learning a model, where the techniques of data mining or machine learning are usually employed. Model-based CF is more flexible and effective for the large-scale and sparse dataset and performs the prediction more accurately. More information about memory and model-based CF can also be seen in Su and Khoshgoftaar (2009).

The CF methods based on latent factor models (Agarwal & Chen, 2009; Chen et al., 2017; Cheng, Ding, Zhu, & Kankanhalli, 2018; Jhamb & Fang, 2017; Luo, Sun, Wang, Li, & Shang, 2017; Luo, Zhou, Li, & Shang, 2017; Sun, Guo, Zhang, & Xu, 2017) characterize users and items in a specific latent space, and describe the rating by the corresponding inner product. One popular latent factor model is the Singular Value Decomposition (SVD), which alleviate the sparsity problem, represents users and items by the vectors in $\mathbb{R}^s$ and computes the rating by vector inner product, and is widely adopted

* Corresponding author at: Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907, USA.
*E-mail addresses:* gu129@purdue.edu (Y. Gu), yang1023@purdue.edu (X. Yang), peng245@purdue.edu (M. Peng), guanglin@purdue.edu (G. Lin).

https://doi.org/10.1016/j.eswa.2019.112885
0957-4174/

in rating prediction (Yang, Yu, Liu, Nie, & Wang, 2016). Biased SVD (Koren, 2008) improved SVD by introducing user and item bias terms. SVD++ (Koren, 2008; Koren & Bell, 2015) extended the biased SVD model by integration of implicit feedback information to get more accurate results/enhance accuracy. Some other techniques use the information of not only the ratings matrix but also the user/item-content. Auto SVD/SVD++ (Zhang, Yao, & Xu, 2017; Zhang, Yao, Xu, Wang, & Zhu, 2017) generalized contractive autoencoder paradigm (Rifai, Vincent, Muller, Glorot, & Bengio, 2011) into SVD/SVD++, hence leverage implicit user feedback and make more accurate recommendations. CFSVD-TF (Wang, Han, Miao, & Zhang, 2019/05) found the implicit feature space by SVD and built the model by introducing trust factors. Another two-level SVD algorithm (Cui et al., 2018) improved the performance by using time information and utilizing direct SVD on the ratings matrix.

In a multi-class rating environment (e.g. scored rating between 1.0 and 5.0), the SVD-type latent factor models are essentially performing low-rank decomposition on the incomplete user-item ratings matrix by solving a model optimization. Practically, the loss function has a great many local extreme points that impede the use of standard optimization solvers with random initial guesses. It motivates us to design a specifical initial guess to accelerate the optimization process

On the other hand, the existing of noise is quite common in raw ratings. For example, irrational or disruptive users may score negatively on commonly good items. Another possible reason is some rating scores are not only from users' intrinsic preference but also the influence of the RS on the users (Sinha, Gleich, & Ramani, 2016). In this point, it can also be claimed not all the ratings are equally meaningful and important to the RS. Hence, the raw ratings collected through the user interface are often with noise and may not reflect all users' interest truthfully, which fails to maintain the significant low-rank property. The traditional SVD-type models rely on the low-rank assumption on the ratings matrix and therefore might perform unsatisfactorily on the highly noisy data. Hence we are encouraged to distinguish users/items by weighting technique, strengthening the influence of information from "good" users/items.

In this work, we first review some existing SVD-type models and then introduce an approach to generate decent initial guesses for the optimization process (Section 2). The strategy is to applying sparse SVD solvers to the normalized ratings matrix. Moreover, to reduce the impact of noise and unreliable ratings, we modify the SVD model by adding non-negative weights to each existed rating entry according to their importance of maintaining the low-rank property (Section 3). The choice of weights can be either user-based or user-item-based. All the proposed models are tested on three public accessible datasets (MovieLens 100k, MovieLens 1M and MovieTweetings) in comparison with biased SVD, SVD++, AutoSVD and AutoSVD++ (Section 4) The experiments show the proposed weighted models outperform other models in improving computational efficiency, obtaining more accurate predictions, and its robustness for noised data. The discussion and conclusion are presented in the end (Section 5).

Compared with previous models in this direction, our contributions of this paper are summarized as follows:

- The starting point of iteration influences the computation efficiency and accuracy for the prediction of SVD-type models. To the best of our knowledge, this work is the first attempt to provide approach for initial point computation. The proposed initial guess computation method in Section 2 helps to start the iteration from an appropriate initial guess and attains a lower local minimum within a fixed number of iterations, hence improves the computational efficiency.
- The proposed weighted SVD-type latent factor models add three different types of weights to each existed rating according to their significance of maintaining the low-rank property that the SVD method relies on, leading to further improvements in prediction accuracy.
- As noise frequently occurs in rating data, the proposed weighted models leverage the bad influence of noise and unreliable ratings, and thus naturally providing prediction more robust to noise than previous work.

## 2. SVD-type latent factor models

### 2.1. Model description

#### 2.1.1. SVD

The SVD-type model is one of the basic latent factor models which follow the idea of truncated singular value decomposition in matrix computation. Let $U$ and $I$ be the total number of users and items in the current dataset, and all users (items) are indexed from 1 to $U(I)$. Denote $r_{ui}$ be the rating score given by user $u$ on the item $i$. If the user $u$ has not rated item $i$, we say $r_{ui}$ is unknown. Let $\mathcal{K}$ be the set consisting of all pairs $(u, i)$ such that $r_{ui}$ is known. Now $\{r_{ui}\}$ forms a ratings matrix by taking users as rows and items as columns. For most practical dataset, majority of the user-item ratings matrix is filled with unknown entries, which need to be predicted. The predicted value for $r_{ui}$ is denote by $\hat{r}_{ui}$. Also, let $s$ be a positive integer representing the number of latent factors. The main framework of the (basic) SVD model is characterizing $r_{ui}$ by the inner product of a user-related $s$-vector $p_u$ and a item-related $s$-vector $q_i$, or

$$\tilde{r}_{ui} = p_u^T q_i. \tag{1}$$

where $p_u, q_i \in \mathbb{R}^s$ are determined by the following optimization problem,

$$\min_{p_u, q_i} J_{\text{SVD}} := \sum_{(u,i)\in\mathcal{K}} (r_{ui} - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2), \tag{2}$$

for $1 \le u \le U$, $1 \le i \le I$. And $\lambda > 0$ is a regularization constant avoiding overfitting.

If gradient-type methods are utilized to solve the optimization, one may need the following expressions of gradients,

$$\frac{\partial J_{\text{SVD}}}{\partial p_{u'}} = 2 \sum_{(u',i)\in\mathcal{K}} \left(-(r_{u'i} - p_{u'}^T q_i) q_i + \lambda p_{u'}\right), \tag{3}$$

$$\frac{\partial J_{\text{SVD}}}{\partial q_{i'}} = 2 \sum_{(u,i')\in\mathcal{K}} \left(-(r_{ui'} - p_u^T q_{i'}) p_u + \lambda q_{i'}\right). \tag{4}$$

When obtaining $p_u$ and $q_i$, the predicted ratings $\hat{r}_{ui}$ can be evaluated by (1).

#### 2.1.2. Biased SVD

Moreover, if the data is shifted by the overall average rating

$$\mu := \sum_{(u,i)\in\mathcal{K}} r_{ui}/|\mathcal{K}|, \tag{5}$$

and scalar baselines $b_u$ and $b_i$ are attached to user $u$ and item $i$ in the SVD model, then we directly have the following biased SVD model (Koren, 2008)

$$\tilde{r}_{ui} = \mu + b_u + b_i + p_u^T q_i. \tag{6}$$

where $b_u$, $b_i$, $p_u$, $q_i$ are determined by the following optimization problem,

$$\min_{b_u, b_i, p_u, q_i} J_{\text{bSVD}} := \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + \|b_u\|^2 + \|b_i\|^2). \tag{7}$$

*2.1.3. SVD++*

The model can be further improved by introducing users' implicit feedback. The most basic criteria for implicit feedback is whether a user gives a rating to a specific item, regardless of the rating score. By virtue of such information, the SVD++ model (Koren, 2008) can be formulated by

$$\tilde{r}_{ui} = \mu + b_u + b_i + q_i^T\left(p_u + |\mathcal{K}_u|^{-\frac{1}{2}}\sum_{(u,j)\in\mathcal{K}} y_j\right). \tag{8}$$

where $\mathcal{K}_u$ is the set consisting of all the items rated by user $u$, and $b_u$, $b_i$, $p_u$, $q_i$, $y_i$ are determined by the following optimization problem,

$$\min_{b_u,b_i,p_u,q_i,y_i} J_{\text{SVD++}} := \sum_{(u,i)\in\mathcal{K}}\left(r_{ui} - \mu - b_u - b_i - q_i^T\left(p_u + |\mathcal{K}_u|^{-\frac{1}{2}}\sum_{(u,j)\in\mathcal{K}} y_j\right)\right)^2 + \lambda\left(\|p_u\|^2 + \|q_i\|^2 + \|b_u\|^2 + \|b_i\|^2 + \|y_i\|^2\right), \tag{9}$$

*2.1.4. AutoSVD, AutoSVD++*

Based on the preceding biased SVD and SVD++, we take advantage of the content of items and extract the corresponding feature vector by contractive auto-encoders (Rifai et al., 2011), then the AutoSVD and AutoSVD++ models (Zhang, Yao, & Xu et al., 2017) can be built as

$$\tilde{r}_{ui} = \mu + b_u + b_i + p_u^T(\beta\cdot \text{CAE}(c_i) + \varepsilon_i), \tag{10}$$

and

$$\tilde{r}_{ui} = \mu + b_u + b_i + \left(p_u + |R(u)|^{-\frac{1}{2}}\sum_{(u,j)\in\mathcal{K}} y_j\right)^T(\beta\cdot \text{CAE}(c_i) + \varepsilon_i), \tag{11}$$

where $\text{CAE}(c_i)$ is the feature vector extracted from item-based content information and $\varepsilon_i$ is an item-based offset vector. Different from SVD, biased SVD and SVD++, which merely take the ratings matrix as the dataset, AutoSVD and AutoSVD++ requires additional information from the data content.

### 2.2. Initial guess computation

For SVD-type models, the loss function is usually a quadratic polynomial for each user-based or item-based vector, and a quartic polynomial for each scalar variable. The non-convexity of the loss function in the admissible set determines the existence of great number of local minima. Taking the SVD model (1)–(2) as an example, since the number of scalar variables is $s(U+I)$, it has theoretically at most $2^{s(U+I)}$ local minima. Starting the gradient-type methods from a decent initial point may accelerate the optimization to a great extent. Actually, many advanced SVD-type models (biased SVD, SVD++ etc.) have a mathematically equivalent loss function to that of the basic SVD model, but have more restricted admissible sets by using different variants of the loss function. Hence performing such SVD-type models can be seen as performing the basic SVD model with different initial guesses. Indeed, starting from an appropriate initial guess will reach to a lower local minimum within a fixed number of iterations, which will also be shown in the experiments. Now we present an approach to find a decent initial guess by using direct sparse SVD solver. For simplicity, we take the SVD model (1)–(2) as an example to explain how to compute good initial $p_u$ and $q_i$. The technique can be slightly modified for biased SVD and SVD++ without difficulty.

We denote the incomplete ratings matrix by

$$\boldsymbol{R} = [r_{ui}]_{u=1,\dots,U}^{i=1,\dots,I} \in \mathbb{R}^{U\times I}, \tag{12}$$

Also, denote

$$\boldsymbol{P} = [p_1\ p_2\ \dots\ p_U] \in \mathbb{R}^{s\times U}, \quad \boldsymbol{Q} = [q_1\ q_2\ \dots\ q_I] \in \mathbb{R}^{s\times I}. \tag{13}$$

Then (1) implies

$$\boldsymbol{R} \approx \boldsymbol{P}^T\boldsymbol{Q} \tag{14}$$

for all $r_{ui}$ where $(u,i)\in\mathcal{K}$. Solving (2) is equivalently looking for rank $s$ approximation of $\boldsymbol{R}$. Let

$$\hat{\boldsymbol{R}} = \begin{cases} r_{ui} - \mu, & (u,i)\in\mathcal{K} \\ 0, & (u,i)\notin\mathcal{K} \end{cases} \tag{15}$$

which is a sparse matrix approximating $\boldsymbol{R}-\mu$ by using 0 to fill the unknown positions, and we do truncated singular value decomposition on $\hat{\boldsymbol{R}}$ by sparse SVD solvers, obtaining

$$\hat{\boldsymbol{R}} \approx \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \tag{16}$$

where $\boldsymbol{U}\in\mathbb{R}^{U\times(s-1)}$, $\boldsymbol{V}\in\mathbb{R}^{I\times(s-1)}$ have orthonormal columns, and $\boldsymbol{S}\in\mathbb{R}^{(s-1)\times(s-1)}$ is diagonal with nonnegative entries. Now let

$$\hat{\boldsymbol{P}}^* := \boldsymbol{S}^{\frac{1}{2}}\boldsymbol{U}^T, \quad \hat{\boldsymbol{Q}}^* := \boldsymbol{S}^{\frac{1}{2}}\boldsymbol{V}^T, \tag{17}$$

then $(\hat{\boldsymbol{P}}^*)^T\hat{\boldsymbol{Q}}^*$ is a good approximation to $\boldsymbol{R}-\mu$. Finally, let

$$\boldsymbol{P}^* = \begin{bmatrix}\sqrt{\mu} & \sqrt{\mu} & \cdots & \sqrt{\mu} \\ & \hat{\boldsymbol{P}}^* & & \end{bmatrix}, \boldsymbol{Q}^* = \begin{bmatrix}\sqrt{\mu} & \sqrt{\mu} & \cdots & \sqrt{\mu} \\ & \hat{\boldsymbol{Q}}^* & & \end{bmatrix}, \tag{18}$$

then $\boldsymbol{P}^*\in\mathbb{R}^{s\times U}$, $\boldsymbol{Q}^*\in\mathbb{R}^{s\times I}$ and $(\boldsymbol{P}^*)^T\boldsymbol{Q}^*$ is close to $\boldsymbol{R}$. Finally we take $p_u^*$ be the $u$th column of $\boldsymbol{P}^*$ and $q_i^*$ be the $i$th column of $\boldsymbol{Q}^*$ to be the initial guess for gradient-type method. This choice allows us to start the searching process from a much smaller loss function than using random generated initial guess, and hence it is more likely to obtain a lower solution within fixed number of iterations.

## 3. Weighted SVD models

### 3.1. User-based weighted model

In a rating dataset from the real-world, all ratings are not equally important. Usually different users provide different contribution to the dataset. For example, movie lovers probably rate movies seriously and frequently, thus are more likely to provide significant data. Normal users rate movies out of their own preference, but are sometimes influenced by external factors. Negative users may give unreasonable or biased ratings, which decrease the reliability of the dataset. This classification of users is also true for items. We regard all the unconvincing ratings as noise added to the raw dataset. Therefore, all users (items) can be regarded differently by their contribution to reliability of the dataset. Weighting strategy can be employed for this goal. Specifically, a user-based scalar weight $w_u$ can be attached to each rating term in the dataset. Hence the loss function of the SVD model (2) can be modified to the following weighted form

$$J_{\text{wSVD}} := \sum_{(u,i)\in\mathcal{K}} w_u(r_{ui} - p_u^Tq_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2). \tag{19}$$

Similarly, the loss function of the biased SVD can be modified to

$$J_{\text{wbSVD}} := \sum_{(u,i)\in\mathcal{K}} w_u(r_{ui} - \mu - b_u - b_i - p_u^Tq_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + \|b_u\|^2 + \|b_i\|^2), \tag{20}$$

which can be referred to as the weighted biased model. A normalization condition is placed to the weights to balance the square error part and regularization part of the loss function, that is

$$\sum_{(u,i)\in\mathcal{K}} w_u = |\mathcal{K}|. \tag{21}$$

### 3.2. Computation of weights

The determination of weights can be based various principles. Specifically, when performing SVD-type models, the basic assumption is the low-rank property of the ratings matrix. Indeed, the ratings matrix from a real-world dataset usually has a low rank, since users of similar interest probably give similar ratings on same items, and items having similar features are more likely to receive similar rating scores from most users. However, some noise, such as casual or biased ratings, may lower the low-rank property, and finally reduce the accuracy of SVD-type models. Hence our strategy of setting weights is to weaken the effect of the users who give "bad" ratings (those far away from preserving the low-rank structure) more frequently.

In practical implementation, we first perform the SVD model (1)–(2) on the original dataset for once, and compute the following entrywise absolute error

$$e_{ui} := |r_{ui} - p_u^T q_i|, \quad (u,i) \in \mathcal{K}. \tag{22}$$

Then the user absolute error can be evaluated by

$$e_u := \left(\sum_{i \in \mathcal{K}_u} e_{ui}\right)/|\mathcal{K}_u|. \tag{23}$$

The user absolute error reflects the degree of preserving the low-rank structure of each user. Out of our strategy, the user having a high user absolute error should be associated with a relatively lower weight, therefore each user-based weight can be computed by

$$\widehat{w}_u = \phi(e_u), \tag{24}$$

$$w_u = (\widehat{w}_u|\mathcal{K}|)/\sum_{(u,i)\in\mathcal{K}} \widehat{w}_u, \tag{25}$$

where $\phi$ is a non-increasing mapping. Note the weights are scaled in (25) according to the normalization condition (21). After obtaining the weights, the weighted models (19) and (20) can be performed and the unknown ratings can be predicted by (1) and (6).

### 3.3. User-item-based weighted model

Actually, the weighting technique can be not only be user-based but also item-based under the principle of low-rank assumption. We can weaken the effect of the items which are more likely to receive "bad" ratings. Denote $\mathcal{K}_i$ by the set consisting of all users who have rated item $i$, then the item absolute error can be evaluated by

$$e_i := \left(\sum_{u \in \mathcal{K}_i} e_{ui}\right)/|\mathcal{K}_i|. \tag{26}$$

and the corresponding item-based weight can be computed by

$$\widehat{w}_i = \phi(e_i). \tag{27}$$

Therefore, each known rating $r_{ui}$ can be equipped with a user-item-based weight $w_{ui}$ which is computed by

$$\widehat{w}_{ui} = \widehat{w}_u \widehat{w}_i, \tag{28}$$

$$w_{ui} = (\widehat{w}_{ui}|\mathcal{K}|)/\sum_{(u,i)\in\mathcal{K}} \widehat{w}_{ui}. \tag{29}$$

Finally, the user-item-based weighted biased model is given by

$$J_{\text{wbSVD}} := \sum_{(u,i)\in\mathcal{K}} w_{ui}(r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + \|b_u\|^2 + \|b_i\|^2), \tag{30}$$

In this model, each known rating is equipped with a weight that depends on both the behavior of the rating user and the property of the rated item.

## 4. Experiments

### 4.1. Datasets and noise setting

Some experiments are implemented to test the performance of the proposed weighted models on three public accessible datasets, MovieLens 100k (ML-100k), MovieLens 1M (ML-1M) and MovieTweetings (MT). MovieLens datasets are collected from the MovieLens website (https://movielens.org) whose rating scores are ranged from 1 to 5, and they are widely employed to assess recommending system techniques in the recent decade. The version of 100k dataset containing about $10^5$ ratings from 943 users on 1660 movies and the 1M dataset including about $10^6$ ratings of approximately 3700 movies made by 6040 users will be used in our experiments. MovieTweetings is a relatively new dataset (https://github.com/sidooms/MovieTweetings) which is collected from Twitter and consisting of about $7.8 \times 10^5$ ratings ranged from 1 to 10 from approximately 53,000 users for 31,000 movies. In practical implementation, we first extract randomly 80% ratings from the datasets to form the training set, and leave the rest to be the testing set. More information about these datasets are summarized in Table 1.

Other than the tests on original real-world datasets, we have implemented tests on datasets with artificial noise to show the robustness of the models. For the preceding three datasets, we place Gaussian noise to each known rating in the training set, obtaining a modified rating score which is

$$(r_{ui})_{\text{noised}} = \min\{\max\{r_{ui} + \xi, M\}, m\} \tag{31}$$

where $\xi \in \mathcal{N}(0, \sigma^2)$ is a Gaussian random noise and $\{M, m\}$ are upper and lower thresholds for all ratings. In the experiments, we set $M = 5.9$, $m = 0.1$ for MovieLens datasets and $M = 10.9$, $m = 0.1$ for MovieTweetings. As we know, the artificial noise will lower the low-rank structure of the ratings matrix. Tests on such noised data will directly reflect the robustness of the models for real-world data which has no significant low-rank property.

### 4.2. Error-weight mapping

When performing the weighted models, the choice of the non-increasing mapping $\phi$ is technical. In the experiments, we take

$$\phi(x) = (M_e^\alpha - x^\alpha)^{1/\alpha}, \tag{32}$$

where $M_e$ is the maximum of all entrywise errors. This definition of the error-weight mapping has the following property.

- Range: all the weights mapped from entrywise errors belongs [0,1];
- Convexity: the parameter $\alpha$ determines the convexity of the mapping. For $0 < \alpha < 1$, $\phi$ is convex; for $\alpha = 1$, $\phi$ is linear; for $\alpha > 1$, $\phi$ is concave.

**Table 1**
Datasets information.

| Datasets | # of users | # of items | # of ratings | Sparcity |
|---|---|---|---|---|
| MovieLens 100k | 943 | 1660 | 99,973 | 6.39% |
| MovieLens 1M | 6040 | 3684 | 1,000,184 | 4.49% |
| MovieTweetings | 52,728 | 30,622 | 780,740 | 0.048% |

**Table 2**
weighted RMSE for various datasets from compared models ($s = 10$).

| Methods | ML-100k | | | ML-1M | | | MT | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\sigma = 0$ | $\sigma = 0.5$ | $\sigma = 1$ | $\sigma = 0$ | $\sigma = 0.5$ | $\sigma = 1$ | $\sigma = 0$ | $\sigma = 0.5$ | $\sigma = 1$ |
| bSVD | 0.93699 | 0.94184 | 0.94886 | 0.90836 | 0.90999 | 0.91396 | 1.42292 | 1.42642 | 1.43684 |
| bSVD* | 0.90677 | 0.92012 | 0.95710 | 0.85540 | 0.86375 | 0.88961 | 1.41645 | 1.42157 | 1.43764 |
| SVD++ | 0.90900 | 0.91902 | 0.94104 | 0.87900 | 0.88170 | 0.88820 | 1.42218 | 1.42787 | 1.43446 |
| SVD++* | 0.90293 | 0.91958 | 0.95960 | 0.86102 | 0.86631 | 0.88538 | 1.40508 | 1.40925 | 1.42183 |
| ASVD | – | – | – | 0.86575 | 0.87349 | 0.90184 | – | – | – |
| ASVD++ | – | – | – | 0.86758 | 0.89245 | 0.96000 | – | – | – |
| $\text{wSVD}_u^*(\alpha = 0.5)$ | 0.87056 | 0.93228 | 0.93228 | 0.81067 | 0.82430 | 0.86165 | 1.34421 | 1.35462 | 1.37630 |
| $\text{wSVD}_u^*(\alpha = 1)$ | 0.87829 | 0.89355 | 0.93876 | 0.83561 | 0.84544 | 0.87627 | 1.38634 | 1.39091 | 1.40698 |
| $\text{wSVD}_u^*(\alpha = 2)$ | 0.89537 | 0.90855 | 0.94751 | 0.84884 | 0.85722 | 0.88448 | 1.41213 | 1.41677 | 1.43655 |
| $\text{wSVD}_{ui}^*(\alpha = 0.5)$ | **0.83902** | **0.86187** | **0.92752** | **0.80131** | **0.81674** | **0.85559** | **1.31671** | **1.32941** | **1.35452** |
| $\text{wSVD}_{ui}^*(\alpha = 1)$ | 0.87352 | 0.89005 | 0.93625 | 0.83241 | 0.84297 | 0.87368 | 1.38138 | 1.38716 | 1.40110 |
| $\text{wSVD}_{ui}^*(\alpha = 2)$ | 0.89401 | 0.90763 | 0.94669 | 0.84821 | 0.85675 | 0.88379 | 1.40104 | 1.40552 | 1.41979 |

The error-weight mapping of various $\alpha$ has different points. The concave mapping with a large $\alpha > 1$ place nearly equally high weights to ratings with small entrywise errors, but much lower weights to those with large entrywise error. It means the concave weighting technique almost does not distinguish good users/items, but tends to greatly abandon the information from bad users/items. Indeed, in real world it is natural that even the users with same interest may give different scores to a particular item, which means two user vectors can hardly be exactly parallel. Hence slight distortion on the low-rank structure is normal and allowed, that the set of good users with small mean entrywise errors do not have to be separated too much. on the contrary, the convex mapping with a small $\alpha < 1$ makes a small part of good ratings highly weighted, but place nearly equally low weights to the rest of ratings. This convex weighting technique aims to mainly rely on the information from the "best" part of users/items, and lower the effect of all the rest.

### 4.3. Algorithm

In the experiments, we implement the proposed weighted biased SVD models with user-based weights (20) and user-item-based weights (30). We denote the testing set by $\mathcal{P}$. The SVD model (2) is performed for once in the beginning to evaluate the entrywise errors. For the optimization process, the sparse SVD technique discussed in Section 2.2 is employed. All procedures are summarized in Algorithm 1.

**Algorithm 1** Algorithm for weighted biased SVD.

**Require:** ratings matrix $\boldsymbol{R}$, $s$, $\lambda$, $\alpha$, $K$
**Ensure:** predicted ratings $\widetilde{r}_{ui}$ for $(u, i) \in \mathcal{P}$
  find initial $P^*$, $Q^*$ by sparse SVD
  **for** $k = 1, \ldots, K$ **do**
    $e_{ui} \leftarrow r_{ui} - p_u^T q_i$
    $p_u \leftarrow p_u + \tau(e_{ui} q_i - \lambda p_u)$, $q_i \leftarrow q_i + \tau(e_{ui} p_i - \lambda q_i)$
  **end for**
  $e_u \leftarrow \left(\sum_{i \in \mathcal{K}_u} e_{ui}\right)/|\mathcal{K}_u|$, $e_i \leftarrow \left(\sum_{u \in \mathcal{K}_i} e_{ui}\right)/|\mathcal{K}_i|$
  $w_u \leftarrow \phi(e_u; \alpha)$, $w_i \leftarrow \phi(e_i; \alpha)$, $w_{ui} \leftarrow (w_u w_i)/\sum_{(u,i) \in \mathcal{K}} w_u w_i$
  compute $\mu$, set $b_u$, $b_i$, find initial $P^*$, $Q^*$ by sparse SVD
  **for** $k = 1, \ldots, K$ **do**
    $e_{ui} \leftarrow r_{ui} - \mu - b_u - b_i - p_u^T q_i$
    $b_u \leftarrow b_u + \tau(w_{ui} e_{ui} - \lambda b_u)$, $b_i \leftarrow b_i + \tau(w_{ui} e_{ui} - \lambda b_i)$
    $p_u \leftarrow p_u + \tau(w_{ui} e_{ui} q_i - \lambda p_u)$, $q_i \leftarrow q_i + \tau(w_{ui} e_{ui} p_i - \lambda q_i)$
  **end for**
  $\widetilde{r}_{ui} \leftarrow \mu + b_u + b_i + p_u^T q_i$ for $(u, i) \in \mathcal{P}$

Note in Algorithm 1, $\tau$ is a suitable step length found by line search process in the gradient-type method.

### 4.4. Evaluation environment and metrics

Experiments are executed on Windows operating system with a 4-core Intel Core(tm) i7-4850K Processor and 32GB of RAM. Biased SVD, SVD++ and weighted biased SVD are performed through Matlab, during which the `svds` routine is utilized to do sparse SVD in the initial guess computation. AutoSVD and AutoSVD++ are performed by the Python codes uploaded to Github[1] (Zhang, Yao, & Xu et al., 2017). Thanks to the sparse data structure, the storage occupied when running the proposed weighted model is no larger than 1G for all the datasets.

Usually the mean error computed from the testing set is taken as a metric to evaluate a recommending system. Since we have distinguish users/items by adding weights in the training process, it is natural to use weighted mean to evaluate the proposed weight-type models. We introduce the following weighted Root Mean Squared Error (RMSE) as the metric for the experiments,

$$\text{RMSE} = \sqrt{\sum_{(u,i) \in \mathcal{P}} w_{ui} (r_{ui} - \tilde{r}_{ui})^2}, \tag{33}$$

where the testing weights $w_{ui}$ can be either used-based or user-item-based depending on which weighted model is performed. They are computed by the same manner as the model weights discussed in Sections 3.2 and 3.3, and obey the following normalization condition,

$$\sum_{(u,i) \in \mathcal{P}} w_{ui} = 1. \tag{34}$$

### 4.5. Overall comparison

In this section, we assess the performance of our proposed models with user-based weights ($\text{wSVD}_u^*$) and user-item-based weights ($\text{wSVD}_{ui}^*$), as well as the existing SVD-type models biased SVD (bSVD), SVD++, AutoSVD (ASVD), AutoSVD++ (ASVD++) for comparison. The regularization parameters $\lambda$ is set to be 0.1 for MovieLens datasets and 0.1 for MovieTweetings. The number of iterations of the gradient descent method is fixed to be 100, after which the weighted RMSE is computed. For weighted models, $\alpha = 0.5, 1, 2$ correspond to the convex, linear and concave error-weight mappings. On noise setting, Gaussian noise with $\sigma = 0.5$ and 1 is added to present original datasets to create lowly noised and highly noised sets.

Tables 2 and 3 show the results of RMSE for various datasets from compared models for $s = 10$ and 20. For these methods, the

[1] https://github.com/cheungdaven/autosvdpp.

**Table 3**
Weighted RMSE for various datasets from compared models ($s = 20$).

| Methods | ML-100k | | | ML-1M | | | MT | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\sigma = 0$ | $\sigma = 0.5$ | $\sigma = 1$ | $\sigma = 0$ | $\sigma = 0.5$ | $\sigma = 1$ | $\sigma = 0$ | $\sigma = 0.5$ | $\sigma = 1$ |
| bSVD | 0.93687 | 0.94185 | 0.94942 | 0.90830 | 0.90995 | 0.91392 | 1.42329 | 1.42679 | 1.43677 |
| bSVD* | 0.90634 | 0.92175 | 0.97665 | 0.85375 | 0.86759 | 0.91016 | 1.41649 | 1.42198 | 1.44177 |
| SVD++ | 0.90780 | 0.91834 | **0.93962** | 0.87958 | 0.88098 | 0.88733 | 1.42481 | 1.42780 | 1.43413 |
| SVD++* | 0.90380 | 0.92392 | 0.98878 | 0.85735 | 0.86532 | 0.88975 | 1.40508 | 1.40929 | 1.42326 |
| ASVD | – | – | – | 0.86365 | 0.87046 | 0.90755 | – | – | – |
| ASVD++ | – | – | – | 0.89162 | 0.93540 | 1.05408 | – | – | – |
| $wSVD^*_u(\alpha = 0.5)$ | 0.85026 | 0.87132 | 0.94866 | 0.81032 | 0.82947 | 0.88379 | 1.34593 | 1.35645 | 1.37897 |
| $wSVD^*_u(\alpha = 1)$ | 0.87809 | 0.89398 | 0.95770 | 0.83458 | 0.84997 | 0.89755 | 1.38854 | 1.39362 | 1.40992 |
| $wSVD^*_u(\alpha = 2)$ | 0.89479 | 0.90904 | 0.96693 | 0.84749 | 0.86135 | 0.90549 | 1.41470 | 1.42057 | 1.44329 |
| $wSVD^*_{ui}(\alpha = 0.5)$ | **0.83985** | **0.86227** | 0.94238 | **0.80071** | **0.82089** | **0.87724** | **1.32081** | **1.33355** | **1.35834** |
| $wSVD^*_{ui}(\alpha = 1)$ | 0.87342 | 0.89021 | 0.95498 | 0.83184 | 0.84701 | 0.89348 | 1.38342 | 1.38919 | 1.40350 |
| $wSVD^*_{ui}(\alpha = 2)$ | 0.89343 | 0.90797 | 0.96612 | 0.84706 | 0.86070 | 0.90360 | 1.40284 | 1.40760 | 1.42285 |

**Table 4**
Computational time (seconds).

| Datasets | ML-100k | ML-1M | MT |
|---|---|---|---|
| bSVD($s = 10$) | 5.80 | 38.93 | 140.20 |
| bSVD($s = 20$) | 5.93 | 44.70 | 149.23 |
| SVD++($s = 10$) | 88.52 | 1505.62 | 984.65 |
| SVD++($s = 20$) | 140.5 | 2477.54 | 1352.99 |
| $wSVD_u(s = 10)$ | 9.34 | 65.40 | 230.75 |
| $wSVD_u(s = 20)$ | 9.97 | 77.32 | 250.60 |
| $wSVD_{ui}(s = 10)$ | 9.09 | 65.55 | 231.65 |
| $wSVD_{ui}(s = 20)$ | 10.23 | 76.42 | 247.95 |

superscript * means the usage of the specifically designed initial guess discussed in Section 2.2 (otherwise the random initial guess is used). Comparing bSVD (SVD++) with bSVD* (SVD++*), the usage of proposed initial guess helps to reduce the RMSE for these methods to some extent. Furthermore, The proposed methods produce better predictions with smaller RMSE than previous methods for all the cases, regardless of various datasets and noisy levels. Specifically, the reduction of RMSE of proposed weighted-type methods is in average 5% than the SVD++ methods and 3% than the bSVD, ASVD and ASVD++ methods. We can also observe from the tables that in these experiments the convex weight mapping works better than the linear and concave ones.

Table 4 compares the computational time for all the models. The comparison of computational time is executed on Matlab platform. All methods are programmed by same logic and data structure. The proposed weighted models take slightly longer while comparable time than the bSVD models, which both greatly outperform the SVD++ models in saving 10 to 100 times computational time.

## 5. Discussion and conclusion

In this work, we develop a modified SVD-type latent factor model for rating prediction in recommending systems. Depending on the error produced by the SVD model, specific weights are computed and added to each entry of the ratings matrix, and a corresponding weighted loss function is built and optimized. Under the low-rank assumption, the entries producing larger errors in the SVD model are equipped with lower weights. In the evaluation of weights, the concave, linear and convex non-increasing mappings are utilized, associated with various emphasis.

In the experiments, the proposed weighted models outperform some existing SVD-type models for original and noised datasets from the real world. For all types of the weighted models, that employs user-item-based weights and convex error-weight mapping performs best, which implies in these datasets, the positive contribution from “good” users/items is more significant than the negative effect caused by bad users/items, and placing more importance on the former is a better strategy. Moreover, the experiments on the artificially noised datasets indicate this weighting technology is robust when dealing with noise. Although the proposed weights are computed based on the low-rank property on the ratings matrix, the noised datasets with less such property can still be handled well by the weighted models.

The main limitation of the weighted models presented in this work is the lack of using the user/item aspect or content. The choice of weights is directly based on the entrywise error produced by the traditional SVD model, hence the effect of the methods highly relies on the effect of the traditional SVD model and the low-rank property of the data. The proposed weighted model may work worse if the ratings matrix has no low-rank property. In this case, taking advantage of the aspect or content feature will greatly improve the effect. Besides, in this work, the weighted RMSE is taken as an evaluation metric that is less convincing in some special cases. More experiments can be implemented by using standard metrics on both real-world and artificially noised datasets.

The proposed weighting technique on RS can be widely studied in the future. In our work, the model weights are labeled in term of users and items. While for datasets with huge number of users and items, adopting clustering approach will greatly improve the efficiency. We suggest the usage of cluster-based weights (Frmal & Lecron, 2017) on CF models of SVD or other types. Moreover, it would be interesting to find various strategies of setting weights, such as activity-based (Salah et al., 2016), aspect-based (Yang et al., 2016), feature-based (Chen et al., 2017) strategies. On the other hand, the weights can be also taken as model variables and determined during the optimization (under some specific assumptions). In this case, the loss function will become highly non-linear (because of the product of weights and latent vectors), that implies the usage of neural network technique on the model instead of usage of straightforward loss functions. Furthermore, it would be interesting and challenging to adjust the weighted models for cold start problem, especially the setup of weighting for new users and items. We anticipate that the weighted method will achieve further accuracy improvement and provide a useful tool to make recommendation.

## Declaration of Competing Interest

None.

## Acknowledgments

We gratefully acknowledge the support from National Science Foundation (DMS-1555072, DMS-1736364 and DMS-1821233).

## References


Agarwal, D., & Chen, B.-C. (2009). Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*. In *KDD '09* (pp. 19–28). New York, NY, USA: ACM. doi:10.1145/1557019.1557029.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence*. In *UAI'98* (pp. 43–52). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Chee, S. H. S., Han, J., & Wang, K. (2001). RecTree: An efficient collaborative filtering method. In Y. Kambayashi, W. Winiwarter, & M. Arikawa (Eds.), *Data warehousing and knowledge discovery* (pp. 141–151). Berlin, Heidelberg: Springer Berlin Heidelberg.

Chen, J., Zhang, H., He, X., Nie, L., Liu, W., & Chua, T.-S. (2017). Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval* (pp. 335–344). ACM.

Cheng, Z., Ding, Y., Zhu, L., & Kankanhalli, M. (2018). Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *Proceedings of the 2018 world wide web conference*. In *WWW '18* (pp. 639–648). Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee. doi:10.1145/3178876.3186145.

Cui, L., Huang, W., Yan, Q., Yu, F. R., Wen, Z., & Lu, N. (2018). A novel context-aware recommendation algorithm with two-level SVD in social networks. *Future Generation Computer Systems, 86*, 1459–1470. doi:10.1016/j.future.2017.07.017.

Delporte, J., Karatzoglou, A., Matuszczyk, T., & Canu, S. (2013). Socially enabled preference learning from implicit feedback data. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 145–160). Springer.

Frmal, S., & Lecron, F. (2017). Weighting strategies for a recommender system using item clustering based on genres. *Expert Systems with Applications, 77*, 105–113. doi:10.1016/j.eswa.2017.01.031.

Gao, H., Tang, J., Hu, X., & Liu, H. (2015). Content-aware point of interest recommendation on location-based social networks. *Twenty-ninth AAAI conference on artificial intelligence*.

Jhamb, Y., & Fang, Y. (2017). A dual-perspective latent factor model for group-aware social event recommendation. *Information Processing & Management, 53*(3), 559–576.

Ju, C., & Xu, C. (2013). A new collaborative recommendation approach based on users clustering using artificial bee colony algorithm. *The Scientific World Journal, 2013*, 869658. doi:10.1155/2013/869658.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*. In *KDD '08* (pp. 426–434). New York, NY, USA: ACM. doi:10.1145/1401890.1401944.

Koren, Y., & Bell, R. (2015). Advances in collaborative filtering. In *Recommender systems handbook* (pp. 77–118). Springer.

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge University Press.

Luo, X., Sun, J., Wang, Z., Li, S., & Shang, M. (2017). Symmetric and nonnegative latent factor models for undirected, high-dimensional, and sparse networks in industrial applications. *IEEE Transactions on Industrial Informatics, 13*(6), 3098–3107.

Luo, X., Zhou, M., Li, S., & Shang, M. (2017). An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications. *IEEE Transactions on Industrial Informatics, 14*(5), 2011–2022.

Marovic, M., Mihokovic, M., Miksa, M., Pribil, S., & Tus, A. (2011). *Automatic movie ratings prediction using machine learning*.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on artificial intelligence* (pp. 187–192). Menlo Park, CA, USA: American Association for Artificial Intelligence.

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on international conference on machine learning*. In *ICML'11* (pp. 833–840). USA: Omnipress.

Salah, A., Rogovschi, N., & Nadif, M. (2016). A dynamic collaborative filtering system via a weighted clustering approach. *Neurocomputing, 175*, 206–215.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on electronic commerce*. In *EC '00* (pp. 158–167). New York, NY, USA: ACM. doi:10.1145/352871.352887.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web*. In *WWW '01* (pp. 285–295). New York, NY, USA: ACM. doi:10.1145/371920.372071.

Sinha, A., Gleich, D. F., & Ramani, K. (2016). Deconvolving feedback loops in recommender systems. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems 29* (pp. 3243–3251). Curran Associates, Inc. http://papers.nips.cc/paper/6283-deconvolving-feedback-loops-in-recommender-systems.pdf.

Su, X., & Khoshgoftaar, T. M. (2006). Collaborative filtering for multi-class data using belief nets algorithms. In *Proceedings of the 18th IEEE international conference on tools with artificial intelligence*. In *ICTAI '06* (pp. 497–504). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ICTAI.2006.41.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence, 2009*. doi:10.1155/2009/421425. 4:2–4:2.

Sun, Z., Guo, G., Zhang, J., & Xu, C. (2017). A unified latent factor model for effective category-aware recommendation. In *Proceedings of the 25th conference on user modeling, adaptation and personalization* (pp. 389–390). ACM.

Walczak, S. (2003). Knowledge-based search in competitive domains. *IEEE Transactions on Knowledge & Data Engineering, 17*(03), 734–743. doi:10.1109/TKDE.2003.1198402.

Wang, J., Han, P., Miao, Y., & Zhang, F. (2019/05). A collaborative filtering algorithm based on SVD and trust factor. *2019 international conference on computer, network, communication and information systems (CNCI 2019)*. Atlantis Press. doi:10.2991/cnci-19.2019.5.

Wang, J., & Ke, L. (2014). Feature subspace transfer for collaborative filtering. *Neurocomputing, 136*, 1–6.

Xu, B., Bu, J., Chen, C., & Cai, D. (2012). An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on world wide web*. In *WWW '12* (pp. 21–30). New York, NY, USA: ACM. doi:10.1145/2187836.2187840.

Yang, C., Yu, X., Liu, Y., Nie, Y., & Wang, Y. (2016). Collaborative filtering with weighted opinion aspects. *Neurocomputing, 210*, 185–196.

Zhang, S., Yao, L., & Xu, X. (2017). Autosvd++: An efficient hybrid collaborative filtering model via contractive auto-encoders. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. In *SIGIR '17* (pp. 957–960). New York, NY, USA: ACM. doi:10.1145/3077136.3080689.

Zhang, S., Yao, L., Xu, X., Wang, S., & Zhu, L. (2017). Hybrid collaborative recommendation via semi-autoencoder. In *International conference on neural information processing* (pp. 185–193). Springer.