Regular Paper

An Extension for Bounded-SVD — A Matrix Factorization Method with Bound Constraints for Recommender Systems

Bang Hai Le[1] Kazuki Mori[2] Ruck Thawonmas[2,a)]

Received: June 30, 2015, Accepted: December 7, 2015

Abstract: In this paper, we introduce a new extension for bounded-SVD, i.e., a matrix factorization (MF) method with bound constraints for recommender system. In bounded-SVD, the bound constraints are included in the objective function so that not only the estimation errors are minimized but the constraints are also taken into account during the optimization process. Our previous results on major real-world recommender system datasets showed that bounded-SVD outperformed an existing MF method with bound constraints, BMF, and it is also faster and simpler to implement than BMF. However, an issue of bounded-SVD is that it does not take into account the bias effects in given data. In order to overcome this issue, we propose an extension of bounded-SVD: bounded-SVD bias. Bounded-SVD bias takes into account the rating biases of users and items – known to reside in recommender system data. The experiment results show that the bias extension can improve the performance of bounded-SVD in most cases.

Keywords: collaborative filtering, matrix factorization, bound constraints, recommender systems, stochastic gradient descent

1. Introduction

Matrix factorization (MF) [1] is one of the state-of-the-art collaborative filtering approaches to recommender systems. In MF, the collected data are formed as a sparse evaluation matrix whose rows and columns correspond to the users and the items, respectively, with each element representing the evaluation (e.g., rating value) of an item of interest made by the corresponding user. In order to recommend unseen items that the users prefer, we need a way to estimate the unknown evaluations of the items. To solve that problem, in MF, the evaluation matrix is estimated as the product of two smaller size matrices, i.e., the user feature matrix and the item feature matrix where the number of latent features is decided in advance. Once these feature matrices are learned, their product can give us the estimations of the unknown evaluations in the original evaluation matrix. MF has been proven to have high prediction performance in many domains of recommender systems while not requiring much additional information about users and items [1].

Many MF based methods have been proposed for recommender system problems over the last decade. In those methods, the main task is to train the user feature matrix and the item feature matrix in order to minimize the estimation errors on the set of known evaluation values. Different methods use different constraints and optimization techniques for training the feature matrices. For example, the SGD method proposed by Funk [2] uses stochastic gradient descent technique for optimization and a constraint which biases the search to the feature matrices with small element values. Bias-SVD [3] and SVD++ [3] (SVD – Singular Value Decomposition) proposed by Koren extend MF with baseline estimation and implicit evaluations, and use gradient descent to learn the feature matrices. Another well-known MF approach is non-negative matrix factorization (NMF) [4] that tries to learn the feature matrices with non-negative elements.

In many recommender system problems, the evaluation values are typically bounded within a range of possible values (e.g., one to five in a five stars rating system). For the estimated evaluations that are out of the range, the methods proposed so far often just simply truncate them to a value within the range. Recently, a new MF method, called Bounded Matrix Factorization (BMF), which takes into account the bound constraints on the evaluation values, was proposed by R. Kannan et al. [5]. In the BMF method, the bounds of the evaluation values are used as a constraint for optimizing the latent feature matrices so that the estimated evaluation values are guaranteed to not go out of the range. As reported by R. Kannan et al., BMF outperforms many state-of-the-art algorithms for recommender systems on major real-world datasets.

However, in the BMF method, in order to guarantee that the estimated evaluation values are bounded within the range, the method has to strictly control the value of each element of the feature matrices. More specifically, for optimizing each element, the method has to compute the lower bound and the upper bound vectors and only updates the element if the new value is bounded by the two bound vectors. This process makes BMF take very long time to finish an iteration, and in many cases, it wastes compu-

[1] Graduate School of Information Science & Engineering, Ritsumeikan University, Kusatsu, Shiga 525–8577, Japan
[2] College of Information Science & Engineering, Ritsumeikan University, Kusatsu, Shiga 525–8577, Japan
[a)] ruck@is.ritsumei.ac.jp

tational time because the new value for the element does not satisfy the bound constraints. In order to solve that problem, in our previous work [6], we introduced a new MF method with bound constraints called bounded-SVD. In this method, the bound constraints are included in the objective function of the problem, so by minimizing the objective function, the method both minimizes the estimation errors and biases the search to the parameter values that satisfy the bound constraints. In addition to having less computational complexity than BMF, bounded-SVD outperformed it according to our previous results on the same datasets used in Kannan et al. [5].

Although bounded-SVD showed promising results on the benchmark datasets compared with BMF, there is still room for improvement. Given an unknown evaluation value, bounded-SVD estimates it only by calculating the product of the corresponding user and item feature vectors. However, it has been shown that in many recommender systems, there are some users who tend to give higher (or lower) evaluation than the average users, and there are also some items that tend to receive higher (or lower) than the average items. Those effects are called user and items biases [1]. The effects of user and item biases were first introduced by Paterek in Ref. [7] and were successfully applied in later works by Koren to improve the performance of MF based models (Refs. [3], [8]). Therefore, in this paper, we introduce an extension of bounded-SVD, namely, bounded-SVD bias that takes into account the user and item biases. The experiment results on major benchmark datasets show that adding biases can improve the performance of bounded-SVD in most cases.

2. Notations

In the following sections, we use the notations shown in **Table 1** for describing the proposed methods.

3. Methodology

In this section, we give an explanation about our previously proposed method, bounded-SVD, and its new extension, bounded-SVD bias.

3.1 Bounded-SVD

For self-containment, we briefly describe bounded-SVD here. In this method, the task is to minimize the estimation errors on the known evaluation values subject to the constraint that the estimated evaluation values are bounded between the minimal and the maximal evaluation values. Therefore, the problem can be written as

$$\min_{P,Q} \sum_{r_{ui} \in \boldsymbol{S}_r} (r_{ui} - \boldsymbol{p}_u^T \boldsymbol{q}_i)^2$$

$$\text{subject to} \quad r_{\min} \leq \boldsymbol{p}_u^T \boldsymbol{q}_i \leq r_{\max}$$

In typical MF methods, in order to avoid overfitting in the training process, a regularization term is often added to the objective function. The most widely used regularization term so far is Frobenius norm [1], whose effect is to penalize the feature matrices with large element values (a sign of overfitting). However, to fulfill the bound constraints, we used a new regularization term that penalizes the feature matrices which produce out-of-bounds estimations. Equation (1) shows the objective function of bounded-SVD with the new regularization term.

Table 1 Notations.

Notations	Meaning
N	Number of users
M	Number of items
K	Number of latent features
$\boldsymbol{R} \in \mathbb{R}^{N\times M}$	Evaluation matrix
$\boldsymbol{P} \in \mathbb{R}^{N\times K}$	User feature matrix
$\boldsymbol{Q} \in \mathbb{R}^{K\times M}$	Item feature matrix
u	A user
i	An item
r_{ui}	The evaluation value of item i made by user u
$\widehat{r_{ui}}$	The estimated evaluation value of item i made by user u
$\boldsymbol{p_u}^T \in \mathbb{R}^{1\times K}$	The feature vector of user u (the u^{th} row of $\boldsymbol{P}$)
$\boldsymbol{q_i} \in \mathbb{R}^{K\times 1}$	The feature vector of item i (the i^{th} column of $\boldsymbol{Q}$)
$\boldsymbol{S_r}$	Set of known evaluation values
$\boldsymbol{S}$	Set of known and unknown evaluation values
r_{min}	Minimal evaluation value
r_{max}	Maximal evaluation value

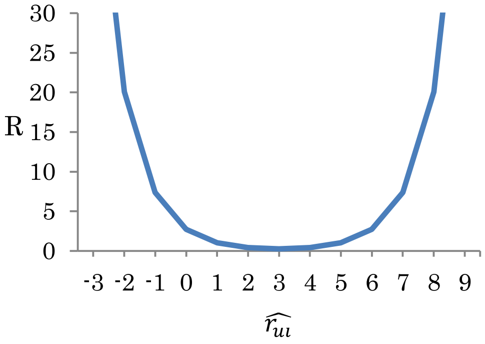

Fig. 1 The value of the regularization term R when $r_{\min} = 1$, $r_{\max} = 5$.

$$E = \sum_{r_{ui} \in \boldsymbol{S}_r} \left(r_{ui} - \boldsymbol{p}_u^T \boldsymbol{q}_i\right)^2 + \left(e^{\left(\boldsymbol{p}_u^T \boldsymbol{q}_i - r_{\max}\right)} + e^{\left(r_{\min} - \boldsymbol{p}_u^T \boldsymbol{q}_i\right)}\right) \qquad (1)$$

In Eq. (1), the value of the regularization term

$$\mathrm{R} = e^{\left(\boldsymbol{p}_u^T \boldsymbol{q}_i - r_{\max}\right)} + e^{\left(r_{\min} - \boldsymbol{p}_u^T \boldsymbol{q}_i\right)}$$

will be very small if the estimation value of r_{ui}, i.e., $\widehat{r_{ui}} = \boldsymbol{p}_u^T \boldsymbol{q}_i$, is inside the range $[r_{\min}, r_{\max}]$. However, when $\boldsymbol{p}_u^T \boldsymbol{q}_i$ goes out of the range, the value of R increases very quickly. **Figure 1** visualizes those effects when $r_{\min}$ and $r_{\max}$ are set to 1 and 5, respectively.

In order to minimize the objective function in Eq. (1), we used a stochastic gradient descent algorithm to optimize the elements of the feature matrices. More specifically, for each known evaluation $r_{ui} \in \boldsymbol{S}_r$, the corresponding feature vectors $\boldsymbol{p}_u^T$, $\boldsymbol{q}_i$ were updated by

$$\boldsymbol{p}_u = \boldsymbol{p}_u + \eta \boldsymbol{q}_i [e_{ui} - H] \tag{2}$$

$$\boldsymbol{q}_i = \boldsymbol{q}_i + \eta \boldsymbol{p}_u [e_{ui} - H] \tag{3}$$

where

- η is the learning rate
- $e_{ui} = r_{ui} - \widehat{r_{ui}} = r_{ui} - \boldsymbol{p}_u^T \boldsymbol{q}_i$
- $H = e^{\left(\boldsymbol{p}_u^T \boldsymbol{q}_i - r_{\max}\right)} - e^{\left(r_{\min} - \boldsymbol{p}_u^T \boldsymbol{q}_i\right)}$

The above process is iterated until the stop criterion (discussed later in the experiments section) is met. Compared with BMF, the time complexity of bounded-SVD in each iteration is much lower, i.e., $\mathbf{O}(K \times |\boldsymbol{S}_r|)$ for bounded-SVD vs $\mathbf{O}(K \times N \times M)$ for BMF.

3.2 Bounded-SVD bias

In bounded-SVD bias, the evaluation values are estimated not only by the product of the feature matrices but also by taking into account the user and item biases. Equation (4) shows the objective function of bounded-SVD bias.

$$E = \sum_{r_{ui} \in \boldsymbol{S}_r} \left(r_{ui} - \bar{r} - b_u - b_i - \boldsymbol{p}_u^T \boldsymbol{q}_i\right)^2 + \left(e^{\left(\bar{r} + b_u + b_i + \boldsymbol{p}_u^T \boldsymbol{q}_i - r_{\max}\right)} + e^{\left(r_{\min} - \bar{r} - b_u - b_i - \boldsymbol{p}_u^T \boldsymbol{q}_i\right)}\right) \tag{4}$$

In Eq. (4), $\bar{r}$ is the average of the known evaluation values while b_u and b_i are the biases of user u and item i, respectively, that will be learned during the training process. As shown in the objective function, each estimated evaluation value is calculated not only by the combination of the corresponding user and item feature vectors but also by taking into account the user and item biases:

$$\widehat{r_{ui}} = \bar{r} + b_u + b_i + \boldsymbol{p}_u^T \boldsymbol{q}_i \tag{5}$$

To minimize the objective function, we also use the stochastic gradient descent algorithm as in the bounded-SVD method. That is, for each known evaluation $r_{ui} \in \boldsymbol{S}_r$, the corresponding feature vectors $\boldsymbol{p}_u^T$, $\boldsymbol{q}_i$ and the bias terms b_u, b_i were updated by:

$$\boldsymbol{p}_u = \boldsymbol{p}_u + \eta \boldsymbol{q}_i [e_{ui} - H] \tag{6}$$

$$\boldsymbol{q}_i = \boldsymbol{q}_i + \eta \boldsymbol{p}_u [e_{ui} - H] \tag{7}$$

$$b_u = b_u + \eta [e_{ui} - H] \tag{8}$$

$$b_i = b_i + \eta [e_{ui} - H] \tag{9}$$

where

- η is the learning rate
- $e_{ui} = r_{ui} - \widehat{r_{ui}} = r_{ui} - \bar{r} - b_u - b_i - \boldsymbol{p}_u^T \boldsymbol{q}_i$
- $H = e^{\left(\widehat{r_{ui}} - r_{\max}\right)} - e^{\left(r_{\min} - \widehat{r_{ui}}\right)}$

Similar to bounded-SVD, the training process is iterated until the stop criterion is met.

4. Implementation

We implemented bounded-SVD and its extension in C++ based on a C++ implementation of GraphChi [11] – an open source disk-based framework for graph computation. GraphChi can run very fast on large-scale recommender system datasets with millions of ratings while not requiring much computational resources.

Table 2 Datasets information.

Dataset	*Users*	*Items*	*Ratings (1e6)*	*Density*	*Ratings range*
Jester	73,421	100	4.1	0.5584	[−10,10]
Movielens	71,567	10,681	10	0.0131	[1,5]
Dating	135,359	168,791	17.3	0.0007	[1,10]
Book	278,858	271,379	1.1	0.00001	[1,10]

5. Experiments

5.1 Experiment Settings

In order to compare the performance of bounded-SVD and its extension with the existing method BMF, we followed the same experiment framework used in Ref. [5]. The datasets we used for the experiments are Jester [10], MovieLens 10M [11], Online dating [12] and Book crossing [13]. **Table 2** shows the characteristics of each dataset. The density value measures the sparseness of each dataset, i.e., the lower density value, the higher sparseness of the dataset. The values for the number of latent features (K) used in the experiments were 10, 20 and 50. Given a dataset, for each setting of K, we prepared 5 experiment data groups. Each experiment data group contains a training set, a validation set and a test set whose data were mutually exclusive and randomly chosen from the whole dataset. We followed the experiment setting reported in Ref. [5] for choosing the proportion of the training, validation and test set with 85%, 5% and 10%, respectively. All methods were run on the same prepared experiment data groups.

The final result for each setting of K was the average root mean square error (RMSE) on the test sets of the corresponding 5 runs. The RMSEs were computed without truncating the estimated ratings as shown in Eq. (10). In this formula, $\boldsymbol{T}_r$ is the set of known ratings of a validation/test set and $n(\boldsymbol{T}_r)$ is the number of known ratings in $\boldsymbol{T}_r$.

$$\sqrt{\frac{\sum_{r_{ui} \in \boldsymbol{T}_r} \left(r_{ui} - \widehat{r_{ui}}\right)^2}{n(\boldsymbol{T}_r)}} \tag{10}$$

In each experiment run, we decided to stop updating the feature matrices once the RMSE on the validation set starts increasing or decreases with too minor amount, i.e., lower than 1e-5.

5.2 Parameters Initialization

In MF methods, a reasonable initialization of the feature matrices is very important for the algorithm to converge to a good solution and there is no exception for our methods. Therefore, in the experiments, we tried with a random and a baseline initialization method that were proposed in Ref. [5]. Below are the descriptions of the two initialization methods.

5.2.1 Random Initialization

The random initialization method initializes the feature matrices $\boldsymbol{P}$, $\boldsymbol{Q}$ such that $\boldsymbol{PQ} \in [r_{\min}, r_{\max}]$. The steps for random initialization are as follow:

- *Step* 1: Initialize $\boldsymbol{P}, \boldsymbol{Q}$ as nonnegative random matrices.
- *Step* 2: Calculate max*Element* that is the maximum value of the elements of $\boldsymbol{PQ}$ without the first column of $\boldsymbol{P}$ and the first row of $\boldsymbol{Q}$.
- *Step* 3: Set $\alpha = \sqrt{\frac{r_{\max}-1}{\max Element}}$
- *Step* 4: Modify $\boldsymbol{P}$ and $\boldsymbol{Q}$ by

$$\boldsymbol{P} = \alpha \boldsymbol{P}$$

$$\boldsymbol{Q} = \alpha \boldsymbol{Q}$$

- *Step* 5: Set the first column of $\boldsymbol{P}$ and the first row of $\boldsymbol{Q}$ to 1's.

In bounded-SVD bias, the estimation formula takes into account the average of known evaluations and the bias terms, so the random initialization procedure for bounded-SVD bias was a bit different. Actually, the variable α in Step 3 was set to $\sqrt{\frac{r_{\max}-\bar{r}-1}{\max Element}}$, and the bias terms were initialized by zeroes.

5.2.2 Baseline Initialization

The idea of baseline initialization is based on the baseline estimator that was presented in Ref. [3]. In the baseline estimator, a missing evaluation r_{ui} is estimated by the following formula:

$$\widehat{r_{ui}} = \bar{r} + b_u + b_i$$

In this formula, $\bar{r}$ is the average of the known evaluations, and b_u, b_i are the bias of user u and item i respectively. In order to make similar estimations with the baseline estimator, in baseline initialization, the feature matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ are initialized by

$$\boldsymbol{P} = \begin{bmatrix} \frac{\bar{r}}{K-2} & \cdots & \frac{\bar{r}}{K-2} & g_1 & 1 \\ \vdots & & \vdots & \vdots & \vdots \\ \frac{\bar{r}}{K-2} & \cdots & \frac{\bar{r}}{K-2} & g_N & 1 \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \\ h_1 & h_2 & \cdots & h_M \end{bmatrix}$$

Namely, for matrix $\boldsymbol{P}$, the first $(K-2)$ columns are set to $\frac{\bar{r}}{K-2}$, the $(K-1)^{th}$ column contains the estimated biases of the users, i.e., $\{g_u | u \in [1, N]\}$, and the elements of the last column are set to 1's. For matrix $\boldsymbol{Q}$, the first $(K-1)$ rows are set to 1's, and the last row contains the estimated biases of the items, i.e., $\{h_i | i \in [1, M]\}$. By that way, given a missing evaluation r_{ui}, its initialized estimation is $\widehat{r_{ui}} = \bar{r} + g_u + h_i$.

For bounded-SVD bias, similar to random initialization, we also needed to modify the procedure of baseline initialization. More specifically, the first $(K-2)$ columns of matrix $\boldsymbol{P}$ were set to zeroes, and the bias terms were initialized by zeroes.

5.3 Tuning the Learning Rate

In this section, we discuss how to choose the values for the learning rate η. In our experiments, the four benchmark datasets differed in the number of users, items and ratings as well as the rating range. Therefore, for each dataset, an appropriate learning rate was needed to allow the algorithm to converge. In order to find an appropriate learning rate for each dataset, we used the grid search method with the set of candidate values for the learning rate being {0.0001, 0.0005, 0.001, 0.005, 0.01}. For each combination of dataset, method, number of latent feature and experiment run, the learning rate value that gave the best performance on the validation sets (lowest RMSE) was chosen as the most appropriate learning rate, and the corresponding trained feature matrices were used to calculate the RMSE on the test sets. In each experiment run, in order to make the algorithm converge, the learning rate slightly decayed by 0.95 after each iteration.

Table 3 The performance of bounded-SVD and bounded-SVD bias compared with BMF.

Dataset	K	Random initialization			Baseline Initialization		
		BMF	bounded-SVD	bounded-SVD bias	BMF	bounded-SVD	bounded-SVD bias
Jester	10	4.6289	4.2078	**4.2041**	4.3320	4.2388	**4.2152**
Jester	20	4.7339	4.1992	**4.1988**	4.3664	4.2438	**4.1592**
Jester	50	4.7180	4.2085	**4.2058**	4.3664	4.1886	**4.1764**
Movielens 10M	10	0.8974	0.8225	**0.8180**	0.8531	0.8335	**0.8293**
Movielens 10M	20	0.8931	0.8246	**0.8191**	0.8526	0.8220	**0.8194**
Movielens 10M	50	0.8932	0.8358	**0.8297**	0.8553	**0.8189**	0.8212
Dating	10	2.1625	1.9758	**1.8871**	1.9309	1.8885	**1.8828**
Dating	20	2.1617	1.8987	**1.8791**	1.9337	1.8899	**1.8881**
Dating	50	2.1642	1.8869	**1.8748**	1.9434	1.8902	**1.8892**
Book Crossing	10	2.8137	2.2796	**1.7430**	1.9355	**1.7399**	1.7411
Book Crossing	20	2.4652	1.9923	**1.7482**	1.9315	**1.7438**	1.7454
Book Crossing	50	2.1269	1.7958	**1.7597**	1.9405	**1.7480**	1.7482

Table 4 The performance of bounded-SVD and bounded-SVD bias with statistical test results.

Dataset	*K*	**Random initialization**		**Baseline Initialization**	
		bounded-SVD	bounded-SVD bias	bounded-SVD	bounded-SVD bias
Jester	10	4.2035	**4.2021**	4.2336	**4.2198**
Jester	20	4.2001	**4.1994**	4.2027	**4.1632 ****
Jester	50	4.2083	**4.2056 ***	4.1937	**4.1700**
Movielens 10M	10	0.8233	**0.8210 ***	0.8314	**0.8290 ****
Movielens 10M	20	0.8245	**0.8190 ****	0.8200	**0.8190 ***
Movielens 10M	50	0.8361	**0.8274 ****	**0.8193 ****	0.8217
Dating	10	1.9602	**1.8804 ****	1.8860	**1.8816 ****
Dating	20	1.9153	**1.8794 ****	1.8892	**1.8881 ***
Dating	50	1.9035	**1.8745 ****	1.8900	**1.8893**
Book Crossing	10	2.2371	**1.7443 ****	**1.7417 ****	1.7426
Book Crossing	20	2.0121	**1.7520 ****	**1.7441 ****	1.7452
Book Crossing	50	1.8011	**1.7591 ****	**1.7480 ****	1.7482

5.4 Experiment results

Table 3 shows the average RMSEs of bounded-SVD and its extension on the benchmark datasets compared with the original BMF method, the lower RMSE the better. The bold numbers are the best RMSE for the corresponding initialization method. We followed the same experiment settings reported in Ref. [5] so the results of BMF were re-used for comparison.

We also conducted statistical tests in order to see how significant the difference between bounded-SVD and bounded-SVD bias is, given the same initialization method. We performed one-sided Wilcoxon signed-rank tests with 20 experiment data groups (5 groups used so far plus 15 additionally generated ones) for each setting of dataset and K. **Table 4** shows the average performance of each method and the significance of each test (* for p-value < 0.05 and ** for p-value < 0.01).

First, from Table 3, we can see that bounded-SVD and bounded-SVD bias outperformed BMF given the same number of latent features and the same initialization method in all benchmark datasets.

Second, the experiment results reported in Table 3 and the statistical test results reported in Table 4 show that given the same initialization method, the performance of bounded-SVD bias was better than bounded-SVD in most cases. More specifically, with random initialization, bounded-SVD bias outperformed bounded-SVD with statistical significance in all benchmark datasets with the same number of latent features except for the Jester dataset with $K = 10$ and 20. In the case of baseline initialization, the performance of bounded-SVD bias was better than bounded-SVD for the Jester, MovieLens 10M and Online dating datasets in all but one case, including five cases with statistical significance, but for the Book-Crossing dataset, bounded-SVD was better with statistical significance.

Third, for the original BMF method, it was clear that given the same number of latent features, the baseline initialization method always gave better performance than the random initialization method. However, that was not the case for bounded-SVD and bounded-SVD bias.

6. Discussions

As we have reported in Section 5.4, bounded-SVD and bounded-SVD bias outperform BMF in all benchmark datasets. One explanation for this is the difference in how the bound constrains are taken into account during the training process. In BMF, the bound constrains are used as hard constraints and are required to be satisfied at any time during the training process. More specifically, BMF will not update a parameter if the bound constraints are not satisfied, even if the update would make the errors smaller. Therefore, BMF can be stuck and make no progress when there is no new update that can satisfy the bound constraints. However, in bounded-SVD and its extension, the bound constraints are used as soft constraints, so it is acceptable for an update to be executed even if it slightly breaks the bound constraints but would make significant performance improvement. Therefore, the training processes of bounded-SVD and its extension are more flexible than BMF and they can find the parameters that contribute to better results.

The difference in the way the bound constraints are utilized can also explain the second result reported in Section 5.4. That is, in BMF, the hard bound constraints have narrowed the search space and therefore, the performance of BMF depends more on the initialization method compared to bounded-SVD and bounded-SVD bias.

Compared to bounded-SVD, bounded-SVD bias could deal with the bias effects in the data and give better performance as we expected, with statistical significance in all but two cases for random initialization, However, for baseline initialization, there were four cases in which bounded-SVD was better with statistical significance. This might be due to some specific characteristics of the dataset, and we would like to find out more about them in future.

7. Conclusions

In this paper, we extended our previous method, i.e., bounded-SVD, by taking into account the user and item biases in recommender system data. The experiment results on major benchmark datasets showed that the extended method, i.e., bounded-SVD bias, gives better performance than bounded-SVD in most cases, and that both bounded-SVD and bounded-SVD bias outperform the existing BMF method. In addition, we discussed why bounded-SVD and its extension give better results than BMF, and how the initialization method affects the performance of each method.

The current version of bounded-SVD and its extension only use the known evaluations for training and have not utilized the context information like users' age, item category, etc. Recently, context-aware recommender systems have become increasingly popular, and it has been shown that the use of context information can significantly improve the performance of recommender system. Therefore, in future work, we would like to improve our methods by utilizing the context information. In addition, we also would like to come up with an automatic way for tuning the learning rate in order to further improve the performance.

Acknowledgments This work was supported in part by Grant-in-Aid for Scientific Research (C), Number 26330421, JSPS.

References

[1] Koren, Y., Bell, R. and Volinsky, C.: Matrix factorization techniques for recommender systems, *Computer*, Vol.42, No.8, pp.30–37 (2009).

[2] Funk, S.: Netflix Update: Try This at Home (Dec. 2006), available from ⟨http://sifter.org/˜simon/journal/20061211.html⟩.

[3] Koren, Y.: Factorization meets the neighborhood: A multifaceted collaborative filtering model, *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.426–434 (2008).

[4] Cichocki, A. and Anh-Huy, P.: Fast local algorithms for large scale nonnegative matrix and tensor factorizations, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.92, No.3, pp.708–721 (2009).

[5] Kannan, R., Ishteva, M. and Park, H.: Bounded matrix factorization for recommender system, *Knowledge and Information Systems*, Vol.39, No.3, pp.491–511 (2014).

[6] Le, B.H., Nguyen, K.Q. and Thawonmas, R.: Bounded-SVD: A Matrix Factorization Method with Bound Constraints for Recommender Systems, *Proc. 2nd International Research Conference on Emerging Information Technology and Engineering Solutions* (*EITES 2015*), Pune, India, pp.23–26 (2015).

[7] Paterek, A.: Improving regularized singular value decomposition for collaborative filtering, *Proc. KDD Cup and Workshop*, Vol.2007, pp.5–8 (2007).

[8] Koren, Y.: Collaborative filtering with temporal dynamics, *Comm. ACM*, Vol.53, No.4, pp.89–97 (2010).

[9] Kyrola, A., Blelloch, G.E. and Guestrin, C.: GraphChi: Large-Scale Graph Computation on Just a PC, *OSDI*, Vol.12, pp.31–46 (2012).

[10] Goldberg, K.: Jester collaborative filtering dataset (2003), available from ⟨http://ieor.berkeley.edu/˜goldberg/jester-data/⟩

[11] Movielens dataset, available from ⟨http://grouplens.org/datasets/movielens/⟩

[12] Brozovsky, L. and Petricek, V.: Recommender system for online dating service, arXiv preprint cs/0703042 (2007).

[13] Ziegler, C.N., McNee, S.M., Konstan, J.A. and Lausen, G.: Improving recommendation lists through topic diversification, *Proc. 14th International Conference on World Wide Web*, pp.22–32 (2005).

Bang Hai Le received an M.Eng. degree in Human Information Science in September 2015 from Ritsumeikan University. His research interests include recommender systems and game AI. He is now with Kaopiz Software Co., Ltd., Vietnam.

Kazuki Mori is a 4th year student at the Department of Human and Computer Intelligence, College of Information Science and Engineering, Ritsumeikan University. His research focuses on recommender systems.

Ruck Thawonmas is a full professor at the College of Information Science and Engineering, Ritsumeikan University. His research interests include game AI, automatic content generation, and player behavior analysis.