AdaError: An Adaptive Learning Rate Method for Matrix Approximation-based Collaborative Filtering

Dongsheng Li
IBM Research - China
Shanghai, China
ldsli@cn.ibm.com

Chao Chen
IBM Research - China
Shanghai, China
cchao@cn.ibm.com

Qin Lv
University of Colorado Boulder
Boulder, CO, USA
qin.lv@colorado.edu

Hansu Gu
Seagate Technology
Longmont, CO, USA
guhansu@gmail.com

Tun Lu
Fudan University
Shanghai, China
lutun@fudan.edu.cn

Li Shang
University of Colorado Boulder
Boulder, CO, USA
li.shang@colorado.edu

Ning Gu
Fudan University
Shanghai, China
ninggu@fudan.edu.cn

Stephen M. Chu
IBM Research - China
Shanghai, China
schu@cn.ibm.com

ABSTRACT

Gradient-based learning methods such as stochastic gradient descent are widely used in matrix approximation-based collaborative filtering algorithms to train recommendation models based on observed user-item ratings. One major difficulty in existing gradient-based learning methods is determining proper learning rates, since model convergence would be inaccurate or very slow if the learning rate is too large or too small, respectively. This paper proposes AdaError, an adaptive learning rate method for matrix approximation-based collaborative filtering. AdaError eliminates the need of manually tuning the learning rates by adaptively adjusting the learning rates based on the noisiness level of user-item ratings, using smaller learning rates for noisy ratings so as to reduce their impact on the learned models. Our theoretical and empirical analysis shows that AdaError can improve the generalization performance of the learned models. Experimental studies on the MovieLens and Netflix datasets also demonstrate that AdaError outperforms state-of-the-art adaptive learning rate methods in matrix approximation-based collaborative filtering. Furthermore, by applying AdaError to the standard matrix approximation method, we can achieve statistically significant improvements over state-of-the-art collaborative filtering methods in both rating prediction accuracy and top-N recommendation accuracy.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Recommender systems**;

KEYWORDS

collaborative filtering, matrix approximation

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186155

ACM Reference Format:
Dongsheng Li, Chao Chen, Qin Lv, Hansu Gu, Tun Lu, Li Shang, Ning Gu, and Stephen M. Chu. 2018. AdaError: An Adaptive Learning Rate Method for Matrix Approximation-based Collaborative Filtering. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3178876.3186155

1 INTRODUCTION

Matrix approximation (MA) methods have become increasingly popular among existing collaborative filtering (CF)-based solutions due to their superior accuracy [3, 4, 8, 9, 15, 16, 22, 44, 46]. In MA-based CF algorithms, gradient-based learning methods such as stochastic gradient descent (SGD) are widely adopted to learn MA models based on observed user-item ratings [8, 16, 22, 43, 44]. The learned MA models are then used to predict user ratings on unseen items. One major difficulty in existing gradient-based learning methods is determining proper learning rates for gradient descent [17], since the model would diverge if the learning rate is too large and the model convergence would be very slow if the learning rate is too small.

Recently, several adaptive learning rate methods, such as Adagrad [12], AdaDelta [45], and Adam [20], have been proposed to address the learning rate issue, and have achieved good performance in several algorithms, especially neural networks [11, 20, 45]. In general, existing adaptive learning rate methods aim to improve model convergence on sparse data by increasing gradient updates for infrequent parameters and decreasing gradient updates for frequent parameters. However, in real-world recommender systems, the observed user-item ratings are not only very sparse but also very noisy [7, 10, 26, 43]. A recent study [10] showed that only 60% of user ratings are unchanged when users are asked to re-rate the same items, and such rating noises can lead to 40% variation in recommendation RMSE [1]. Therefore, it is important to consider rating noises when choosing the learning rates in MA-based CF solutions, i.e., performing small gradient updates for noisy ratings to prevent the learned models from overreacting to rating noises.

To this end, this paper proposes AdaError — an adaptive learning rate method for matrix approximation-based collaborative filtering. AdaError reduces the learning rates for noisy training examples so that the learned models are less prone to the noisy ratings in the training data. AdaError also adaptively shrinks the learning rates as the number of epochs increases, thus eliminating the need of manually tuning the learning rates. Our theoretical and empirical analysis shows that AdaError can improve the generalization performance of learned MA models and are less sensitive to L_2 regularization coefficients. Experimental studies using the MovieLens and Netflix datasets demonstrate that AdaError outperforms state-of-the-art adaptive learning rate methods in matrix approximation-based collaborative filtering. Furthermore, by applying AdaError to the standard matrix approximation method, we can statistically significantly improve recommendation accuracy for both the rating prediction task and the top-N recommendation task, compared with state-of-the-art collaborative filtering methods.

2 PROBLEM FORMULATION

In this section, we first introduce the basic concepts of matrix approximation-based collaborative filtering. Then, we analyze the noisy rating problem in real-world recommender systems. At last, we motivate the targeted problem through a case study.

2.1 Matrix Approximation-based Collaborative Filtering

Given a user-item rating matrix $R \in \mathbb{R}^{m \times n}$, where m is the number of users and n is the number of items, matrix approximation methods aim to determine a user feature matrix $U \in \mathbb{R}^{m \times k}$ and an item feature matrix $V \in \mathbb{R}^{n \times k}$, such that

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

k, the rank of R, is typically much smaller than m and n in real-world recommender systems. After obtaining U and V, the predicted rating of the i-th user on the j-th item can be computed by the dot product of their corresponding feature vectors, i.e., $\hat{r}_{i,j} = U_i V_j^T$.

To obtain optimal U and V in Equation 1, gradient-based learning methods such as stochastic gradient descent (SGD) can be adopted to minimize the following regularized least square error problem [8, 22, 34, 44]:

$$L = \sum_{i,j \in \Omega} (R_{i,j} - U_i V_j^T)^2 + \mu ||U||_F^2 + \mu ||V||_F^2, \tag{2}$$

where Ω is the set of observed entries in the rating matrix R and $||\cdot||_F$ is the Frobenius norm. When using SGD to solve the minimization problem above at entry $R_{i,j}$, the gradient update rules can be described as follows:

$$U_i \leftarrow U_i - \lambda \frac{\partial L}{\partial U_i}, \; V_j \leftarrow V_j - \lambda \frac{\partial L}{\partial V_j}. \tag{3}$$

λ is the learning rate, which controls the convergence of the model learning process.

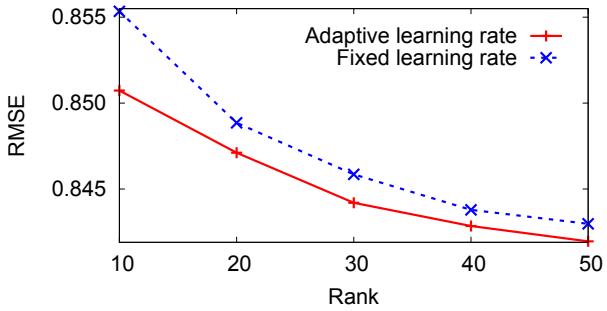

Figure 1: Case study: Recommendation accuracy comparison between RSVD [34] with adaptive / fixed learning rate when varying rank values on the MovieLens 1M dataset. We use 0.01 as the fixed learning rate. For adaptive learning rate, we use 0.011 for entries with small training error and 0.009 for entries with large training error.

2.2 Noisy Ratings in Recommender Systems

User-item ratings in real-world recommender systems are typically noisy [2, 28, 33]. A recent work [33] pointed out that *natural noise*, which arises when recommender systems collect or infer user preferences, commonly exist in today's recommender system databases. The natural noises are caused by various reasons, including difficulty for users to quantify their preferences [18], inappropriate granularity of rating scales [10], memory loss due to the long time between seeing and rating items [30], bad mood [2], etc. These natural noises in user-item ratings are inevitable due to so many complex reasons. As such, matrix approximation-based CF methods should consider rating noises in their algorithm design.

The fraction of noisy ratings in recommender systems is large according to recent studies [10, 18]. Cosley et al. [10] found that about 40% of user ratings are different from their previous ratings when users are asked to re-rate the same movies they rated before. Similarly, Jones et al. [18] found that user stability on rating items is around 63% in their study. These noisy ratings can significantly influence the accuracy of matrix approximation-based CF methods [1, 10]. Cosley et al. [10] observed significant MAE differences when using collaborative filtering algorithms on users' original ratings and new ratings. Amatriain et al. [1] found that the recommendation RMSE variations can be as high as 40% when noises exist in the user-item rating matrix. Therefore, if we can properly address the noisy ratings issue, it is very promising for us to improve the recommendation accuracy of collaborative filtering methods.

2.3 Case Study: Noise vs. Learning Rate

Here, we conduct a case study to empirically investigate the potential improvement in recommendation accuracy when considering rating noises in the model learning process. The study is carried out on the MovieLens 1M dataset, which

contains $\sim 10^6$ ratings from 6k users on 4k items. The basic idea is to use larger learning rates for robust ratings and smaller learning rates for noisy ratings, thus reducing the impact of noisy ratings on the learned model. However, rating noises are difficult to quantify. So we rely on the learned MA models to identify noisy ratings. Specifically, we assume that a rating is noisy if the learned MA model cannot fit the rating accurately, i.e., a rating is noisy if the learned MA model has a large training error on this rating.

Based on the idea above, given a predefined learning rate λ and a training example $R_{i,j}$, we adopt the following adaptive learning rate: 1) if the training example's prediction error $(R_{i,j} - \hat{R}_{i,j})^2$ is larger than the average square error of all training examples, then its learning rate is decreased by 10%, i.e., 0.9λ; 2) otherwise, its learning rate is increased by 10%, i.e., 1.1λ. As shown in Figure 1, RSVD [34] with adaptive learning rate outperforms RSVD with fixed learning rate in recommendation accuracy, i.e., achieving lower Root Mean Square Error (RMSE). And this is true for rank variations from 10 to 50. The only difference between the two methods is that RSVD with adaptive learning rate can use smaller updates for entries with large training error (noisy ratings) and larger updates for entries with small training error (robust ratings). As a result, the learned MA models are less prone to noises in the ratings.

This case study confirms that we can improve the recommendation accuracy of MA-based CF methods by considering rating noises in the model learning process. The key question is – how to design an intelligent adaptive learning rate method which can address the different levels of noises across different ratings for MA-based CF solutions.

3 ALGORITHM DESIGN

In this section, we first propose the AdaError method, which can adaptively adjust the learning rates for entries with different noise levels. Then, we present how to apply the proposed AdaError method in matrix approximation for two collaborative filtering tasks: rating prediction and top-N recommendation.

3.1 The Proposed AdaError Method

AdaError is designed based on the following idea: Entries with larger training errors should be given smaller learning rates and entries with smaller training errors should be given larger learning rates. This idea may be implemented in different ways. Here, we propose an adaptive method which is similar to AdaGrad [12]. Given a predefined learning rate λ and an observed entry $R_{i,j} \in \Omega$, its learning rate at the t-th iteration is defined as follows:

$$\lambda_{i,j}^{(t)} = \frac{\lambda}{\sqrt{E_{i,j}^{(t-1)} + \epsilon}} + \beta. \tag{4}$$

$E_{i,j}^{(t-1)} = \sum_{x=0}^{t-1} (R_{i,j} - \hat{R}_{i,j}^{(x)})^2$ is the sum of squared training error w.r.t. $R_{i,j}$ up to the $(t-1)$-th iteration. ϵ is a small constant to prevent 0 in the denominator, which is set to $1e$-8 in this paper. β is a constant to prevent $\lambda_{i,j}^{(t)}$ from becoming infinitely small after a large number of iterations, which is set to $1e$-4 in this paper.

The advantages of the proposed AdaError method are summarized as follows:

- *Addressing different levels of noises*: The learning rates of different entries will vary according to their training errors, so that entries with different cumulative training errors will have varying learning rates;
- *Adaptive tuning of the learning rates*: The learning rates will shrink as the number of iterations increases, so that it is unnecessary to manually tune the learning rates. Meanwhile, the adoption of β can prevent the learning rates from becoming infinitely small, so that the learning process will stop within an acceptable number of iterations; and
- *Efficiency*: The proposed AdaError method is entry-wise, i.e., different entries will have different learning rates. This is different from many existing parameter-wise adaptive learning rate methods, e.g., AdaGrad [12], AdaDelta [45] and Adam [20], in which the learning rates are different across different parameters. For matrix approximation, the computation complexity of AdaError, which is $O(|\Omega|)$ per iteration, is smaller than that of those parameter-wise adaptive learning rate methods, which is $O(k|\Omega|)$ per iteration.

3.2 AdaError for Rating Prediction

Here, we present the algorithm design of applying the proposed AdaError method for matrix approximation in the rating prediction task, in which we solve the minimization problem defined by Equation 2. We first initialize the parameters and $E \in \mathbb{R}^{m \times n}$. Then, we iteratively update the parameters and $E^{(t)}$ $(t > 0)$ until convergence using stochastic gradient descent. The details are presented in Algorithm 1.

Algorithm 1 AdaError for Rating Prediction

Require: User-item rating matrix R, observed entry set Ω, rank k, learning rate λ, regularization coefficient μ, $\epsilon = 1e-8$, $\beta = 1e-4$.
Ensure: Approximated user-item rating matrix $\hat{R}$.

```
Initialize U, V randomly, t = 1, and E^(0) = 0_{m,n}.
while not converged do
    for each (i, j) ∈ Ω do
        E_{i,j}^(t) ← E_{i,j}^(t−1) + (R_{i,j} − U_i V_j^T)^2.
        λ_{i,j}^(t) ← λ / sqrt(E_{i,j}^(t) + ε) + β.
        U_i ← U_i − 2λ_{i,j}^(t) ((U_i V_j^T − R_{i,j}) V_j + μ U_i).
        V_j ← V_j − 2λ_{i,j}^(t) ((U_i V_j^T − R_{i,j}) U_i + μ V_j).
        t ← t + 1.
    end for
end while
return R̂
```

3.3 AdaError for Top-N Recommendation

Here, we present the algorithm design of applying the proposed AdaError method for matrix approximation in the

top-N recommendation task. In many top-N recommendation tasks, the user-item ratings are binary, i.e., $R_{i,j} \in \{-1, 1\}$. The mean square loss as defined in Equation 2 will not be appropriate in such setting [42]. Therefore, we adopt the "0-1" loss with exponential surrogate function. Note that other surrogate loss functions [42], such as square loss, log loss, and hinge loss, can also be adopted in Equation 5. However, many real-world datasets only provide positive feedbacks, e.g., click through data, which is also known as implicit feedback data. To address the implicit feedback issue, we give a larger weight to positive ratings and a smaller weight to negative ratings following the WRMF method [16]. We set $w_{i,j} = 1$ if $R_{i,j} = 1$ and $w_{i,j} = 0.04$ if $R_{i,j} = -1$ based on our empirical study. Finally, the loss function for top-N recommendation can be defined as follows:

$$L' = \sum_{(i,j)\in\Omega} w_{i,j} \exp\{-R_{i,j}\hat{R}_{i,j}\} + \mu||U||_F^2 + \mu||V||_F^2. \quad (5)$$

Then, we can similarly solve the above minimization problem as in the rating prediction problem. The details are presented in Algorithm 2.

Algorithm 2 AdaError for Top-N Recommendation

Require: User-item rating matrix R, observed entry set Ω, rank k, learning rate λ, regularization coefficient μ, $\epsilon = 1e-8$, $\beta = 1e-4$.
Ensure: Approximated user-item rating matrix $\hat{R}$.

```
Initialize U, V randomly, t = 1, and E^(0) = 0_{m,n}.
while not converged do
  for each (i, j) ∈ Ω do
    E_{i,j}^(t) ← E_{i,j}^(t−1) + w_{i,j} exp{−R_{i,j} U_i V_j^T}.
    λ_{i,j}^(t) ← λ / sqrt(E_{i,j}^(t) + ε) + β.
    U_i ← U_i + λ_{i,j}^(t) (w_{i,j} R_{i,j} V_j exp{−R_{i,j} U_i V_j^T} − 2μ U_i).
    V_j ← V_j + λ_{i,j}^(t) (w_{i,j} R_{i,j} U_i exp{−R_{i,j} U_i V_j^T} − 2μ V_j).
    t ← t + 1.
  end for
end while
return R̂
```

4 THEORETICAL ANALYSIS

In this section, we first analyze the convergence rate of applying AdaError in SGD for solving strongly convex problems. Then, we analyze the generalization error bound of AdaError-based SGD, and compare it with standard SGD.

4.1 Convergence Rate

The convergence rate of SGD has been extensively studied in the literature and a recent result showed that the SGD algorithm can return a solution which is $O(1/T)$-close to the optimum after T iterations [14]. More formally, the convergence rate of SGD can be analyzed as follows [14]:

THEOREM 4.1. *Assuming that the loss function L is l-strongly convex and its gradients satisfy that $\mathbb{E}(||g||^2) \leq G^2$ for all model $w \in \Phi$, where $\mathbb{E}(g) = \nabla L(w)$. Then, there exists a deterministic algorithm that can return a w after at most T iterations such that, for optimal $w^* \in \Phi$, we have $\mathbb{E}[L(w)] - L(w^*) \leq O(\frac{G^2}{lT})$.*

Following the above results, we can similarly derive the convergence rate of AdaError-based SGD in the following Theorem 4.2.

THEOREM 4.2. *Assuming that the loss function L is l-strongly convex and its gradients satisfy that $\mathbb{E}(||g||^2) \leq G^2$ for all model $w \in \Phi$, where $\mathbb{E}(g) = \nabla L(w)$. Then, by properly choosing λ and β in Equation 4 for each iteration, AdaError-based SGD can return a w after at most T iterations such that, for optimal $w^* \in \Phi$, we have $\mathbb{E}[L(w)] - L(w^*) \leq O(\frac{G^2}{lT})$.*

Theorem 4.2 proves that solving a strongly convex loss using AdaError-based SGD can achieve a convergence rate of $O(1/T)$. It is easy to verify that mean square loss (Equation 2) is strongly convex, because its second order derivative is a constant 2. The exponential loss (Equation 5) is not always strongly convex, because its second order derivative $e^x \to 0$ when $x \to -\infty$. However, $-R_{i,j}\hat{R}_{i,j}$ will not diverge to $-\infty$ in matrix approximation if the learning rate is properly set in SGD, so we can assume that $-R_{i,j}\hat{R}_{i,j} \geq C$ for all $(i, j) \in \Omega$ with some properly chosen learning rate. Then, Equation 5 will satisfy the strongly convex assumption.

4.2 Generalization Error Bound

The generalization performance of matrix approximation would be poor if the learned MA models are prone to the noisy training data. Since AdaError can prevent the learned MA models from overreacting to the noises in the training data, AdaError can naturally improve the generalization performance. Here, we theoretically analyze the generalization error bound of AdaError-based SGD.

Uniform stability [6] is adopted to analyze the generalization error of SGD. The expected generalization error of SGD with fixed learning rate can be bounded as follows [13]:

THEOREM 4.3. *Given a loss function $L : \Phi \to \mathbb{R}$, assuming $L(\cdot; x)$ is convex, $||\nabla L(\cdot; x)|| \leq P$ and $||\nabla L(w; x) - \nabla L(w'; x)|| \leq b||w - w'||$ for all training example $x \in X$ and any two models $w, w' \in \Phi$. Suppose that we run SGD with the t-th step size $\lambda \leq 2/b$ for totally T steps. Then, SGD satisfies uniform stability on samples with n examples by $\epsilon_{stab} \leq \frac{2P^2}{n} \sum_{t=1}^{T} \lambda$.*

Following the above results, we can similarly derive the generalization error bound of AdaError-based SGD in the following Theorem 4.4.

THEOREM 4.4. *Given a loss function $L : \Phi \to \mathbb{R}$, assuming $L(\cdot; x)$ is convex, $||\nabla L(\cdot; x)|| \leq P$ and $||\nabla L(w; x) - \nabla L(w'; x)|| \leq b||w - w'||$ for all training example $x \in X$ and any two models $w, w' \in \Phi$. Suppose that we run AdaError-based SGD with the t-th step size as defined in Equation 4 satisfying $\lambda^{(t)} \leq 2/b$ for totally T steps. Then, AdaError-based SGD satisfies uniform stability on samples with n examples by $\epsilon_{stab} \leq \frac{2P^2}{n} \sum_{t=1}^{T} \lambda^{(t)}$.*

PROOF. The proof can be derived from Theorem 4.3. □

Next, we can compare the generalization error bound of SGD with fixed learning rate and AdaError-based SGD in the following Theorem 4.5.

THEOREM 4.5. *The uniform stability bound of Theorem 4.4 will be sharper than that of Theorem 4.3 if* $\sum_t \frac{1}{T\sqrt{E^{(t)}+\epsilon}} \leq 1-\beta/\lambda$.

PROOF. The sharper uniform stability bound of Theorem 4.4 indicates that $\frac{2P^2}{n}\sum_{t=1}^{T}\lambda^{(t)} \leq \frac{2P^2}{n}\sum_{t=1}^{T}\lambda$, i.e., $\sum_{t=1}^{T}\lambda^{(t)} \leq \sum_{t=1}^{T}\lambda$. Based on Equation 4, we know that $\sum_{t=1}^{T}(\lambda/\sqrt{E^{(t)}+\epsilon}+\beta) \leq \sum_{t=1}^{T}\lambda$. Then, by simple algebra, we can complete the proof. □

In AdaError, $E^{(t)}$ will accumulate as the number of iterations increases, so the above condition can be easily met when T is large enough, e.g., $T > 100$ will be fair enough in our empirical studies.

5 EXPERIMENTS

This section first presents the experimental setup. Then, we analyze the sensitivity of AdaError and compare it with other adaptive learning rate methods. At last, we compare the recommendation accuracy of AdaError-based matrix approximation method with state-of-the-art CF methods in both rating prediction and top-N recommendation tasks.

5.1 Experimental Setup

Dataset Description. The following real-world datasets are used in the experiments: 1) MovieLens 100K dataset ($\sim 10^5$ ratings from 1,000 users on 1,700 movies); 2) MovieLens 1M dataset ($\sim 10^6$ ratings from 6,000 users on 4,000 movies); 3) MovieLens 10M dataset ($\sim 10^7$ ratings from 72,000 users on 10,000 movies); and 4) Netflix Prize dataset ($\sim 10^8$ ratings from 480,000 users on 17,770 movies). In each experiment, we randomly split the dataset into training and test sets and keep the ratio as 90% : 10%. All reported results are averaged over five different rounds of random splits. Note that, for top-N recommendation, we predict whether a user will rate an item [19], i.e., the user rating on an item will be 1 if the user rated the item and -1 otherwise.

Parameter Setting. We set $\lambda = 0.01$ and $\beta = 1e\text{-}4$ in Equation 4 if not explicitly specified. The regularization coefficient μ is set to 0.02 for rating prediction and 0.001 for top-N recommendation. The convergence threshold is set to $1e\text{-}5$ and the maximum number of epochs is set to 1000. The optimal parameters of the compared methods are chosen from their original papers.

Compared Methods. For the rating prediction task, we compare the proposed method with the following state-of-the-art MA-based CF methods: 1) BPMF [38] is a Bayesian extension of the PMF [39] method, which can automatically control model capacity by integrating over all model parameters and hyperparameters; 2) DFC [27] can improve the scalability and accuracy of matrix factorization by a divide and conquer-based ensemble strategy; 3) LLORMA [24] is an ensemble MA method, which integrates a set of localized MA models through kernel smoothing; 4) GSMF [44] can model multiple user behaviors through group sparsity regularization in matrix factorization; 5) WEMAREC [8] is also an ensemble method, which integrates biased co-clustering-based MA models by weighted average; 6) SMA [26] can improve the stability of matrix approximation by introducing hard-predictable terms in the loss function; and 7) ERMMA [25] can minimize the expected risk in learning MA models.

For the top-N recommendation task, we compare the proposed method with the following methods: 1) WRMF [16] assigns point-wise confidences to individual ratings in the user-item rating matrix to address the implicit feedback issue; 2) BPR [36] learns a pair-wise loss to optimize ranking measures for top-N setting. They proposed different versions of BPR methods, and this paper compares with the BPR-MF; 3) AOBPR [35] improves the original BPR method by oversampling informative pairs to speed up convergence and accuracy; and 4) SLIM [31] generates top-N recommendations by aggregating weighted user ratings learned by solving an L_1 and L_2 regularized optimization problem.

In addition, we compare AdaError with the following popular adaptive learning rate methods: 1) AdaGrad [12] can adjust the learning rates so that frequently updated parameters will be given smaller learning rates and infrequently updated parameters will be given larger learning rates; 2) RMSprop[1] divides the learning rate with a running average of the magnitudes of recent gradients to prevent the learning rates from becoming infinitely small; and 3) Adam [20] adjusts the learning rates of individual parameters by considering the first moment and the second moment of the gradients.

Evaluation Metrics. For the rating prediction task, root mean square error (RMSE) is adopted to measure recommendation accuracy: $\text{RMSE}(\hat{R}) = \sqrt{1/|\Omega'|\sum_{(i,j)\in\Omega'}(R_{i,j}-\hat{R}_{i,j})^2}$, where Ω' is the set of entries in the test set. Lower RMSE indicates higher rating prediction accuracy. For the top-N recommendation task, two popular measures are adopted: 1) Precision@N $= |I_r \cap I_u|/|I_r|$, where I_r is the list of top-N recommendations and I_u is the list of items that u has rated; 2) NDCG@N = DCG@N/IDCG@N, where DCG@N $=\sum_{i=1}^{n}(2^{rel_i}-1)/log_2(i+1)$ and IDCG@N is the value of DCG@N with perfect ranking ($rel_i = 1$ if u rated the i-th recommended item and $rel_i = 0$ otherwise). Higher Precision@N and NDCG@N indicate higher recommendation accuracy.

5.2 Generalization Error Analysis

Figure 2 compares the gap between training and test errors of RSVD [34] using standard SGD and RSVD with AdaError on the MovieLens 10M dataset. As we can see from the results, RSVD with standard SGD has a much larger gap between training and test errors than that of RSVD with AdaError,

[1] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

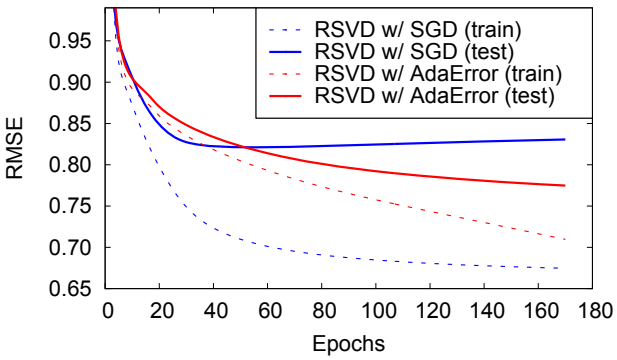

Figure 2: Training and test errors vs. epochs of RSVD [34] using standard SGD and AdaError on the MovieLens 10M dataset. We set rank $k = 100$ and L_2 regularization coefficient $\mu = 0.02$ for both methods.

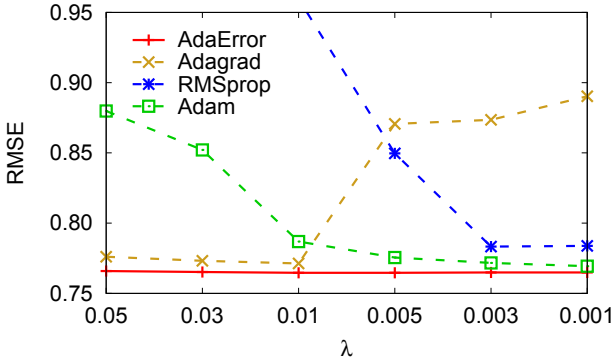

Figure 3: Sensitivity analysis with initial learning step λ on the MovieLens 10M dataset. We set rank $k = 100$ and L_2 regularization coefficient $\mu = 0.02$ for all methods, and set $\beta = 1e-4$ for AdaError.

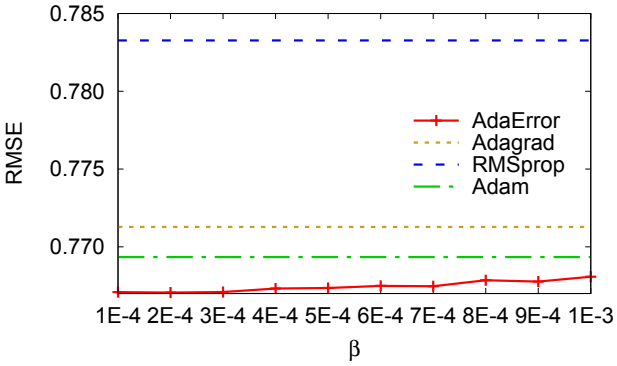

Figure 4: Sensitivity analysis of AdaError with hyperparameter β on the MovieLens 10M dataset. We set rank $k = 100$ and L_2 regularization coefficient $\mu = 0.02$ for all methods.

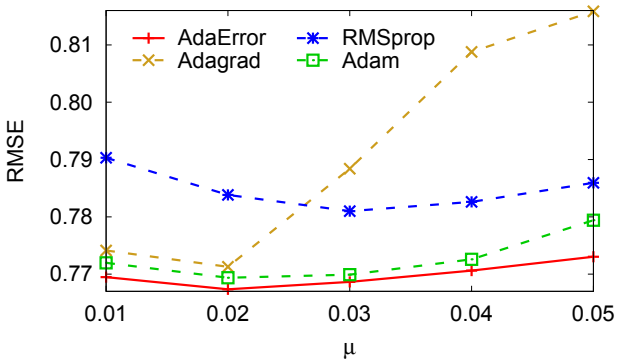

Figure 5: Sensitivity analysis with L_2 regularization coefficient μ on the MovieLens 10M dataset. We set $k = 100$ and $\lambda = 0.01$ for all methods, and set $\beta = 1e-4$ for AdaError.

i.e., RSVD with AdaError can achieve better generalization performance, which confirms the theoretical analysis in Theorem 4.4 and Theorem 4.5 that AdaError can achieve sharper uniform stability bound when the number of epochs is large enough.

5.3 Sensitivity Analysis

Here, we analyze the sensitivity of AdaError with different hyperparameters, and compare AdaError with three popular adaptive learning rate methods: AdaGrad [12], RMSProp and Adam [20]. To ensure fair comparison, all methods are applied on RSVD with the same hyperparameters if not explicitly specified.

5.3.1 Sensitivity vs. λ. Figure 3 compares the recommendation accuracy of AdaError and the three other methods with different initial learning rates. We can see from the results that AdaError achieves the lowest RMSE values among all the compared methods with λ values varying from 0.05 to 0.001. Meanwhile, the test RMSE of AdaError only changes slightly when λ changes, while the test RMSEs of the other methods change significantly. This is because AdaError can always converge to local minimum due to its insensitivity to rating noises.

Note that the test RMSE of AdaGrad dramatically increases when λ is too small, which is because the learning rates in AdaGrad will quickly become very small when λ is not large enough and the training process will terminate due to too small gains in optimization accuracy. Meanwhile, the test RMSEs of RMSProp and Adam are very high when the learning rates are too large, which is because too large learning rates may affect the convergence of these two methods. In comparison, AdaError can overcome this issue, because the smallest learning steps of AdaError is bounded by β.

5.3.2 Sensitivity vs. β. As defined in Equation 4, β can prevent the learning steps of AdaError from becoming infinitely small. As shown in Figure 4, smaller β value can achieve

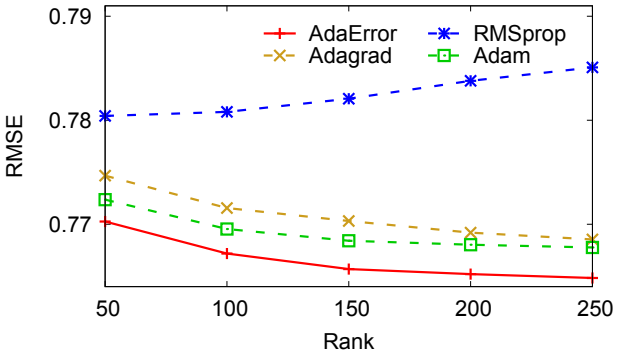

Figure 6: Sensitivity analysis with rank on the MovieLens 10M dataset. The hyperparameters of all the methods are chosen as the optimal ones based on the previous sensitivity analysis.

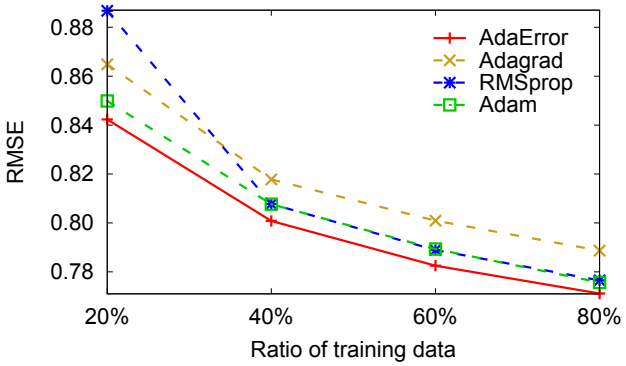

Figure 7: Sensitivity analysis with data sparsity on the MovieLens 10M dataset. We set rank $k = 100$ and L_2 regularization coefficient $\mu = 0.02$ for all methods, and set $\beta = 1e-4$ for AdaError.

slightly lower test RMSE, because smaller β can reduce the overall learning steps of AdaError and smaller learning steps can ensure better convergence around local minimum. However, the test RMSE only increases by approximately 0.001 when β increases from $1e-4$ to $1e-3$, which indicates that AdaError is very stable with different β values.

5.3.3 Sensitivity vs. Regularization Coefficient. Figure 5 compares the recommendation accuracy of AdaError and three other methods with different L_2 regularization coefficients. As we can see from the results, AdaError achieves smaller test RMSE than all the compared methods when μ increases from 0.01 to 0.05. It is known that proper L_2 regularization coefficient can prevent the learned models from overfitting [29]. However, AdaError can naturally prevent the learned models from overfitting the training data, so that AdaError is less sensitive to μ than other methods.

5.3.4 Sensitivity vs. Rank. Figure 6 compares the recommendation accuracy of AdaError and the three other methods with different rank values. Note that, for all the methods, we use the optimal hyperparameters based on the previous sensitivity analysis. As we can see from the results, the test RMSE of AdaError consistently decreases when rank increases from 50 to 250, which indicates that AdaError will not overfit even with very large ranks. This further confirms that AdaError can achieve better generalization performance. Moreover, AdaError outperforms all the other three methods with all ranks, which further confirms that AdaError is more desirable in collaborative filtering.

5.3.5 Sensitivity vs. Data Sparsity. Figure 7 compares the recommendation accuracy of AdaError and three other methods with different training / test split ratios, i.e., different sparsity of training data. As shown in the results, the test accuracy of AdaError consistently outperforms all the three compared methods when the training set ratio increases from 20% to 80%. This indicates that AdaError can achieve superior performance even when the training data is very sparse.

Table 1: RMSE comparison between the proposed method ($k = 500$) and seven state-of-the-art matrix approximation-based CF methods — BPMF [38], DFC [27], LLORMA [24], GSMF [44], WEMAREC [8], SMA [26], ERMMA [25]. Note that the proposed method statistically significantly outperforms the other methods with at least 95% confidence level.

Method	MovieLens (10M)	Netflix
BPMF	0.8197 ± 0.0006	0.8421 ± 0.0003
DFC	0.8067 ± 0.0002	0.8453 ± 0.0003
LLORMA	0.7855 ± 0.0002	0.8275 ± 0.0004
GSMF	0.8012 ± 0.0011	0.8420 ± 0.0006
WEMAREC	0.7775 ± 0.0007	0.8143 ± 0.0001
SMA	0.7682 ± 0.0003	0.8036 ± 0.0004
ERMMA	0.7670 ± 0.0007	0.8018 ± 0.0001
Proposed	**0.7644 ± 0.0003**	**0.7980 ± 0.0002**

In summary, the sensitivity analysis experiments demonstrate that the proposed AdaError method is less sensitive to hyperparameters compared with the three popular adaptive learning rate methods in MA-based collaborative filtering. Therefore, we can conclude that AdaError is more desirable than the other three adaptive learning rate methods in matrix approximation-based collaborative filtering.

5.4 Rating Prediction Accuracy

Table 1 compares the recommendation accuracy of the proposed method (AdaError for rating prediction) with seven state-of-the-art matrix approximation-based collaborative filtering methods on the MovieLens 10M and Netflix datasets. As we can see from the results, the proposed method outperforms all the seven compared methods with at least 95% confidence level on both MovieLens 10M and Netflix datasets. The main reasons are: 1) AdaError can prevent the learned MA models from overreacting to noises, so that the learned

Table 2: Precision comparison between the proposed method and one rating-based MA method (RSVD [34]) and four top-N recommendation methods (WRMF [16], BPR [36], SLIM [31], AOBRP [35]) on the MovieLens 100K and MovieLens 1M datasets.

Metric		Precision@N			
Data | Method		N=1	N=5	N=10	N=20
ML-100K	RSVD	0.3155 ± 0.0038	0.2179 ± 0.0007	0.1403 ± 0.0035	0.1300 ± 0.0057
	WRMF	0.3851 ± 0.0116	0.2752 ± 0.0053	0.2202 ± 0.0056	0.1679 ± 0.0035
	BPR	0.3439 ± 0.0168	0.2533 ± 0.0082	0.2061 ± 0.0040	0.1581 ± 0.0028
	SLIM	0.3951 ± 0.0056	0.2625 ± 0.0090	0.2055 ± 0.0031	0.1539 ± 0.0015
	AOBPR	0.3395 ± 0.0099	0.2591 ± 0.0057	0.2119 ± 0.0031	0.1632 ± 0.0025
	Proposed	**0.4078 ± 0.0021**	**0.2934 ± 0.0049**	**0.2331 ± 0.0029**	**0.1779 ± 0.0018**
ML-1M	RSVD	0.1659 ± 0.0017	0.1263 ± 0.0005	0.1037 ± 0.0009	0.0766 ± 0.0020
	WRMF	0.2761 ± 0.0074	0.2155 ± 0.0009	0.1816 ± 0.0007	0.1459 ± 0.0004
	BPR	0.3062 ± 0.0030	0.2277 ± 0.0074	0.1896 ± 0.0048	0.1516 ± 0.0007
	SLIM	0.3053 ± 0.0097	0.2208 ± 0.0039	0.1836 ± 0.0006	0.1419 ± 0.0029
	AOBPR	0.3098 ± 0.0076	0.2315 ± 0.0002	0.1926 ± 0.0022	0.1540 ± 0.0016
	Proposed	**0.3692 ± 0.0018**	**0.2878 ± 0.0011**	**0.2385 ± 0.0014**	**0.1891 ± 0.0007**

Table 3: NDCG comparison between the proposed method and one rating-based MA method (RSVD [34]) and four top-N recommendation methods (WRMF [16], BPR [36], SLIM [31], AOBRP [35]) on the MovieLens 100K and MovieLens 1M datasets.

Metric		NDCG@N			
Data | Method		N=1	N=5	N=10	N=20
ML-100K	RSVD	0.0389 ± 0.0028	0.1047 ± 0.0032	0.0996 ± 0.0059	0.1393 ± 0.0071
	WRMF	0.0913 ± 0.0034	0.1989 ± 0.0030	0.2535 ± 0.0045	0.3131 ± 0.0043
	BPR	0.0783 ± 0.0036	0.1803 ± 0.0056	0.2351 ± 0.0056	0.2929 ± 0.0065
	SLIM	0.0922 ± 0.0021	0.1967 ± 0.0036	0.2476 ± 0.0050	0.3017 ± 0.0091
	AOBPR	0.0770 ± 0.0043	0.1801 ± 0.0044	0.2343 ± 0.0051	0.2930 ± 0.0058
	Proposed	**0.0998 ± 0.0014**	**0.2143 ± 0.0036**	**0.2719 ± 0.0048**	**0.3333 ± 0.0053**
ML-1M	RSVD	0.0324 ± 0.0020	0.0700 ± 0.0006	0.0864 ± 0.0002	0.1006 ± 0.0001
	WRMF	0.0510 ± 0.0013	0.1202 ± 0.0002	0.1563 ± 0.0013	0.2012 ± 0.0010
	BPR	0.0568 ± 0.0006	0.1235 ± 0.0003	0.1601 ± 0.0035	0.2070 ± 0.0011
	SLIM	0.0551 ± 0.0015	0.1201 ± 0.0023	0.1586 ± 0.0028	0.1948 ± 0.0043
	AOBPR	0.0582 ± 0.0018	0.1200 ± 0.0006	0.1567 ± 0.0009	0.2021 ± 0.0009
	Proposed	**0.0722 ± 0.0010**	**0.1653 ± 0.0006**	**0.2155 ± 0.0002**	**0.2703 ± 0.0003**

MA models can achieve better generalization performance when the hyperparameters of AdaError, i.e., λ and β, are properly chosen and 2) AdaError can shrink the learning rates as the number of iterations increases which can ensure better convergence, because smaller learning steps can reduce oscillation near local minimum.

5.5 Top-N Recommendation Accuracy

Table 2 and Table 3 compare the recommendation accuracy (Precision@N and NDCG@N) between the proposed method and five other methods on the MovieLens 100K and MovieLens 1M datasets, respectively. Among the five compared methods, RSVD [34] is a rating-based MA method and WRMF [16], BPR [36], SLIM [31] and AOBRP [35] are top-N recommendation algorithms. As shown in the results, the proposed method (AdaError for top-N recommendation) outperforms all the compared methods on both Precision@N and NDCG@N when N increases from 1 to 20 with at least 95% confidence level. The main reasons of the superior performance of the proposed method are 1) better generalization performance; 2) stronger capability of reducing oscillation near local minimum; 3) a weighting strategy which gives lower weights to unobserved ratings to address the positive-unlabeled data issue in the top-N recommendation task.

The main difference between RSVD and WRMF is that WRMF can give unobserved ratings lower weights in the training process whereas RSVD treats all ratings equally. This indicates that setting smaller weights for unobserved ratings can significantly improve recommendation accuracy in top-N recommendation on implicit feedback data. The proposed method adopts the same weighting strategy as WRMF, and the superior performance of the proposed method indicates that the proposed AdaError method can improve the performance of weighted matrix approximation in the top-N recommendation task.

6 RELATED WORK

Collaborative filtering is an important class of methods in today's recommender systems and matrix approximation methods are popular among existing CF methods for both rating prediction [8, 22, 25, 26, 38, 44] and top-N recommendation [16, 25, 36]. The earliest matrix approximation-based CF method tried to discover the latent structures within the user-item rating matrix [5], in which they claimed that SVD can eliminate the need for users to rate all similar items. In other words, discovering the latent structures can help alleviate the data sparsity issue in real-world recommender systems [22, 40, 41]. Meanwhile, the scalability of recommender systems can be improved because recommendation scores can be computed by simple dot products of feature vectors [40, 41]. Later, several works have demonstrated that MA methods can achieve superior accuracy than memory-based methods in the Netflix Prize Competition [22, 34], after which MA methods have been the focus of collaborative filtering methods [24]. Salakhutdinov and Mnih first proposed the probabilistic matrix factorization method [39], and they proposed the BPMF method by extending the PMF method using a Bayesian treatment [38]. Koren [21] combined the SVD and neighbor-based CF method and proposed the SVD++ method. Recently, Li et al. [26] proposed a stable matrix approximation method, which can improve the generalization performance of matrix approximation by improving the algorithm stability. Later, they proposed the ERMMA method [25] to reduce the expected risk of MA models. However, the above methods did not consider the noisy rating issue in real-world recommender systems, and the learned MA models in their methods may overfit the noises in the ratings and thus achieve non-optimal accuracy.

The noisy rating issue has been investigated in the literature, and two main categories of works have been proposed to address this issue. The first category of methods tries to alleviate the rating noises by designing new user interfaces or interaction methods [2, 18, 30]. Amatriain et al. [2] proposed to denoise the recommender system databases by asking users to re-rate some of their previously rated items. Nguyen et al. [30] proposed to use exemplars to relate user rating decisions to their prior rating decisions, and they found that presenting exemplars can help users generate more consistent ratings. Jones et al. [18] found that users comparing items are more reliable and stable than rating items as much as 20%, so that they claimed that comparisons could yield better user modeling. However, the above methods require additional efforts for users and thus may not be practical in real-world recommender systems. The other category of methods tries to address the noisy rating issue by designing robust collaborative filtering algorithms [28, 32, 37]. Mehta et al. [28] proposed a robust matrix factorization method based on M-estimator, which can achieve higher accuracy with noisy user profiles. However, their method aimed to make CF more robust when attack profiles are inserted in the databases, which shares the same assumptions with several other approaches [32, 37]. In contrast, our work assumes that rating noises generally exist in the data and users may or may not intentionally insert these noises when rating items, which is more general than the above works. Lakshminarayanan et al. [23] proposed a robust Bayesian matrix factorization method, in which the rating noises are claimed to be non-Gaussian and heteroscedastic. However, their method is not easy to train when the number of parameters is large.

Adaptive learning rate methods have also been proposed to achieve more informative gradient updates than fixed learning rates [12, 20, 45]. Duchi et al. [12] first proposed the AdaGrad method, which can perform larger updates for infrequent parameters and smaller updates for frequent parameters to achieve better convergence. However, their method suffers from the issue that the learning rate will shrink and become infinitely small when the number of iterations is large. To address the above issue, Zeiler [45] proposed the AdaDelta method, in which a running average of the squared gradients is adopted in the learning rate rather than the accumulated squared gradients in AdaGrad. Similarly, RMSProp proposed by Hinton's group adopts the same idea to prevent the learning rates from becoming infinitely small. Recently, Kingma et al. [20] proposed the Adam method, in which the learning rates are adjusted by keeping an exponentially decaying average of both the first moment and the second moment of the gradients. However, the above methods are not intentionally designed for noisy training data, so they are not suitable for learning MA models with noisy ratings. In comparison, our experimental studies show that AdaError is less sensitive to hyperparameters than the above methods, which means that AdaError is more desirable in practice.

7 CONCLUSION

Noisy ratings in real-world recommender systems pose challenges to matrix approximation-based collaborative filtering algorithms, in which the learned MA models will easily be prone to noises in the ratings. This paper proposes AdaError — an adaptive learning rate method for matrix approximation-based collaborative filtering methods. AdaError gives smaller learning rates to ratings with larger noises so as to prevent the learned MA models from overreacting to the noises. Our theoretical analysis shows that AdaError can achieve better generalization performance than fixed learning rate method when the hyperparameters in AdaError are properly chosen. Experimental studies on real-world datasets demonstrate that (1) AdaError can achieve better performance than existing adaptive learning rate methods and (2) our proposed AdaError-based recommendation methods can achieve statistically significant higher recommendation accuracy than state-of-the-art collaborative filtering algorithms in both the rating prediction task and the top-N recommendation task.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61332008 and U1630115, and the National Science Foundation of USA under Grant Nos. 1334351, 1442971, and 1528138.

REFERENCES

[1] Xavier Amatriain, Josep M. Pujol, and Nuria Oliver. 2009. I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP '09)*. Springer, 247–258.

[2] Xavier Amatriain, Josep M. Pujol, Nava Tintarev, and Nuria Oliver. 2009. Rate It Again: Increasing Recommendation Accuracy by User Re-rating. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys '09)*. ACM, 173–180.

[3] Alex Beutel, Amr Ahmed, and Alexander J. Smola. 2015. ACCAM-S: Additive Co-Clustering to Approximate Matrices Succinctly. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. 119–129.

[4] Alex Beutel, Ed H. Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. 2017. Beyond Globally Optimal: Focused Learning for Improved Recommendations. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 203–212.

[5] Daniel Billsus and Michael J Pazzani. 1998. Learning Collaborative Information Filters.. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*, Vol. 98. 46–54.

[6] Olivier Bousquet and André Elisseeff. 2001. Algorithmic Stability and Generalization Performance. In *Advances in Neural Information Processing Systems*. 196–202.

[7] Emmanuel J. Candès and Yaniv Plan. 2010. Matrix Completion With Noise. *Proc. IEEE* 98, 6 (2010), 925–936.

[8] Chao Chen, Dongsheng Li, Yingying Zhao, Qin Lv, and Li Shang. 2015. WEMAREC: Accurate and Scalable Recommendation through Weighted and Ensemble Matrix Approximation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 303–312.

[9] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. 2017. Learning to Recommend Accurate and Diverse Items. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 183–192.

[10] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. 2003. Is Seeing Believing?: How Recommender System Interfaces Affect Users' Opinions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, 585–592.

[11] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*. 1223–1231.

[12] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[13] Moritz Hardt, Benjamin Recht, and Yoram Singer. 2016. Train Faster, Generalize Better: Stability of Stochastic Gradient Descent. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML'16)*. JMLR.org, 1225–1234.

[14] Elad Hazan and Satyen Kale. 2014. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research* 15, 1 (2014), 2489–2512.

[15] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. 2013. Personalized Recommendation via Cross-domain Triadic Factorization. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW '13)*. ACM, 595–606.

[16] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM '08)*. 263–272.

[17] Robert A Jacobs. 1988. Increased rates of convergence through learning rate adaptation. *Neural networks* 1, 4 (1988), 295–307.

[18] Nicolas Jones, Armelle Brun, and Anne Boyer. 2011. Comparisons Instead of Ratings: Towards More Stable Preferences. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT '11)*. IEEE, 451–456.

[19] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored Item Similarity Models for top-N Recommender Systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. ACM, 659–667.

[20] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[21] Yehuda Koren. 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 (KDD '08)*. ACM, 426–434.

[22] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[23] Balaji Lakshminarayanan, Guillaume Bouchard, and Cedric Archambeau. 2011. Robust Bayesian matrix factorisation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 425–433.

[24] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *Proceedings of The 30th International Conference on Machine Learning (ICML '13)*. 82–90.

[25] Dongsheng Li, Chao Chen, Qin Lv, Li Shang, Stephen M. Chu, and Hongyuan Zha. 2017. ERMMA: Expected Risk Minimization for Matrix Approximation-based Recommender Systems. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI '17)*. 1403–1409.

[26] Dongsheng Li, Chao Chen, Qin Lv, Junchi Yan, Li Shang, and Stephen M. Chu. 2016. Low-rank matrix approximation with stability. In *Proceedings of The 33rd International Conference on Machine Learning (ICML '16)*. 295–303.

[27] Lester W Mackey, Michael I Jordan, and Ameet Talwalkar. 2011. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*. 1134–1142.

[28] Bhaskar Mehta, Thomas Hofmann, and Wolfgang Nejdl. 2007. Robust Collaborative Filtering. In *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys '07)*. ACM, 49–56.

[29] Andrew Y. Ng. 2004. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML '04)*. 78–85.

[30] Tien T. Nguyen, Daniel Kluver, Ting-Yu Wang, Pik-Mai Hui, Michael D. Ekstrand, Martijn C. Willemsen, and John Riedl. 2013. Rating Support Interfaces to Improve User Experience and Recommender Accuracy. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, 149–156.

[31] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining (ICDM '11)*. 497–506.

[32] Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guénolé Silvestre. 2004. Collaborative Recommendation: A Robustness Analysis. *ACM Trans. Internet Technol.* 4, 4 (2004), 344–377.

[33] Michael P. O'Mahony, Neil J. Hurley, and Guénolé C.M. Silvestre. 2006. Detecting Noise in Recommender System Databases. In *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI '06)*. ACM, 109–115.

[34] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, Vol. 2007. 5–8.

[35] Steffen Rendle and Christoph Freudenthaler. 2014. Improving Pairwise Learning for Item Recommendation from Implicit Feedback. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM '14)*. 273–282.

[36] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. 452–461.

[37] Paul Resnick and Rahul Sami. 2007. The Influence Limiter: Provably Manipulation-resistant Recommender Systems. In *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys '07)*. ACM, 25–32.

[38] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, 880–887.

[39] Ruslan Salakhutdinov and Andriy Mnih. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[40] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2000. Application of Dimensionality Reduction in Recommender System - A Case Study. In *ACM WebKDD 2000 Workshop*. ACM SIGKDD.

[41] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. In *Proceedings of the 5th International Conference in Computers and Information Technology*.

[42] Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd. 2016. Generalized Low Rank Models. *Foundations and Trends in Machine Learning* 9, 1 (2016), 1–118.

[43] Linli Xu, Zaiyi Chen, Qi Zhou, Enhong Chen, Nicholas Jing Yuan, and Xing Xie. 2016. Aligned Matrix Completion: Integrating Consistency and Independency in Multiple Domains. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 529–538.

[44] Ting Yuan, Jian Cheng, Xi Zhang, Shuang Qiu, and Hanqing Lu. 2014. Recommendation by Mining Multiple User Behaviors with Group Sparsity. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI '14)*. 222–228.

[45] Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

[46] Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma, and Shi Feng. 2013. Localized Matrix Factorization for Recommendation Based on Matrix Block Diagonal Forms. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW '13)*. ACM, 1511–1520.