Contextual Collaborative Filtering via Hierarchical Matrix Factorization

Erheng Zhong *, Wei Fan†and Qiang Yang*

Abstract

Matrix factorization (MF) has been demonstrated to be one of the most competitive techniques for collaborative filtering. However, state-of-the-art MFs do not consider contextual information, where ratings can be generated under different environments. For example, users select items under various situations, such as happy mood vs. sad, mobile vs. stationary, movies vs. book, etc. Under different contexts, the preference of users are inherently different. The problem is that MF methods uniformly decompose the rating matrix, and thus they are unable to factorize for different contexts. To amend this problem and improve recommendation accuracy, we introduce a "hierarchical" factorization model by considering the local context when performing matrix factorization. The intuition is that: as ratings are being generated from heterogeneous environments, certain user and item pairs tend to be more similar to each other than others, and hence they ought to receive more collaborative information from each other. To take the contextual information into consideration, the proposed "contextual collaborative filtering" approach splits the rating matrix hierarchically by grouping similar users and items together, and factorizes each sub-matrix locally under different contexts. By building an ensemble model, the approach further avoids over-fitting with less parameter tuning. We analyze and demonstrate that the proposed method is a model-averaging gradient boosting model, and its error rate can be bounded. Experimental results show that it outperforms three state-of-the-art algorithms on a number of real-world datasets (MovieLens, Netflix, etc). The source code and datasets are available for download[1].

1 Introduction

Recommender systems are often based on collaborative filtering, where we observe m users, n items, and a rating matrix $R = (u_k; i_k; r_k)_{k=1}^{|R|}$ with real-valued ratings r_k. In this formulation, r_k represents the preferences of certain users u_k for some items i_k. The objective is to predict unobserved ratings based on users' past preferences. Among different learning methods, matrix factorization (MF) is generally considered to be a competitive method and there has been much interests since Netflix has accounted its competition winner. Based on announced data, MF models the relationship between users and items by decomposing the rating matrix into low-rank factor matrices and uncovers the latent relationship between users and items. Specifically, the latent factors indicate both users' interest distribution and items' membership over latent topics.

Despite their successful applications, MF methods suffer from one major drawback: the rating values in the matrix are assumed to be generated uniformly, such that a user (or an item) generates all his ratings using the same factor vector, without taking specific contexts into account. In other words, MFs do not consider two facts: (1) a user's ratings can be influenced by multiple factors, such as mood, environment, time of day, etc; (2) items with similar properties would receive similar ratings, such as the ratings of comedy and dramatic movies. To solve the first problem, SVD++ [11] adds constraints from implicit feedbacks on users' factor vectors, and M^{3}F [15] introduces context dependence rating prediction by allowing each item to select a new topic for each new interaction. However, all of these solutions inadvertently introduce heavy computational costs and become rather inefficient when the dataset is large. In addition, none of these approaches solve the second problem.

In this paper, we propose a new matrix factorization model: Random Partition Matrix Factorization (RPMF), based on a tree structure constructed by using an efficient random partition technique adopted from Random Decision Trees (RDT) [4], to provide a fast solution to the problems discussed above. While previous matrix factorization models generate their data uniformly, the proposed contextual RPMF model generate data by region in a hierarchical manner. In other words, the rating values in the matrix generated with either similar contexts or items are partitioned locally unto the same node of a decision tree. Afterwards, we explore low-rank approximation to the current sub-rating-matrix at each node. Therefore, the ratings at tree nodes give rise to different factor vectors to handle specific contexts, and thus ratings with similar items at the same node can provide more impact on each other.

*Hong Kong University of Science and Technology. {ezhong,qyang}@cse.ust.hk.

†IBM T.J.Watson Research Center. weifan@us.ibm.com.

[1] http://www.cse.ust.hk/~ezhong/code/sdm12hmf.zip

Table 1: Definition of notations

Notation	Notation Description
R	Rating matrix
I	Index matrix to indicate non-zeros in R
U	Factor matrix of users
V	Factor matrix of items
m	Number of users
n	Number of items
k	Number of latent factors
F_E	Ensemble Model
N	Number of trees
h	Height of trees
$\{1, \ldots, Q\}$	Rating scope

As such, the latent features implied in the low-rank matrices are explored to partition the rating matrix at the current node. In addition, we take advantage of the efficiency of random partition and its generalizability, and exploit ensemble to effectively reduce the problem of over-fitting. As discussed in Section 4, this process is a model-averaging gradient boosting model, and thus has good performance guarantee. As analyzed, the prediction model is not sensitive to the rank of the latent matrices - a problem associated with many of the state-of-the-art matrix factorization methods. In the prediction process, when a new user-item pair arrives, we pass the pair from the root to the leaf to smooth the predictions at each node on this decision path. We then combine the predictions from each tree in the ensemble together to generate the final prediction. This process is efficient because there are very few iterations required in order for the gradient algorithm to converge towards leaf level. We also demonstrate that its complexity is comparatively smaller than other state-of-the-art methods. A brief summary of the paper is as follows:

1. A novel hierarchical matrix factorization method with three properties: decomposition under different contexts, none over-fitting and low time complexity.
2. A new generalization bound for matrix factorization which can handle multi-class ratings, and its formal analysis to guarantee the performance of the proposed method.
3. Evaluations on real datasets that demonstrate improved performance over state-the-art methods: Yahoo! Music Recommendation, Netflix, Moivelens and Eachmovie.

2 Background and Related Works

We introduce the preliminaries and backgrounds in this section. The notations used in the paper are summarized in Table 1.

2.1 Matrix Factorization Models Matrix factorization methods [11] represent state-of-the-art for collaborative filtering tasks. They learn a constrained decomposition of the rating matrix as a product of two low-rank latent matrices: $R \approx UV^T$ where $R \in \mathbb{N}^{m \times n}$ is the rating matrix, $U \in \mathbb{R}^{m \times d}$ is the latent matrix representing users and $V \in \mathbb{R}^{m \times d}$ is the parallel definition for items. The basic matrix factorization is as follows.

$$\arg\min_{U,V} \parallel (R - UV^T) \odot I \parallel_f^2 + \lambda \cdot \Big(\parallel U \parallel_f + \parallel V \parallel_f \Big) \tag{2.1}$$

where I is the index matrix to indicate the non-zero elements in R, $\odot$ represents element-wise multiplication and $\parallel * \parallel_f$ is the Frobenius norm and used to regulate the complexity of U and V, and λ is the trade-off parameter. In practice, we explore stochastic gradient descent (SGD) to compute U and V. To obtain accurate results, SVD++ [11] integrates the implicit feedback and preference bias:

$$R_{ij} = b_{ij} + \Big(U_i + |N(i)|^{-\frac{1}{2}} \sum_{v \in N(i)} Y_v\Big) V_j^T \tag{2.2}$$

where b_{ij} is the bias for user i on item j, $N(i)$ is the number of implicit feedbacks obtained from user i, and Y_v is the feedback factor vector of item v. However, these methods need to set the rank of the latent matrices, d, carefully to avoid over-fitting. [20] explores Bayesian method to avoid this and uses MCMC to estimate the parameters. To take the advantage of kernel method, [12] proposes a non-linear matrix factorization based on Gaussian process latent variable models (GP-LVM). The common drawback of these methods is that the computation complexity is too high when the dataset is large and makes them impractical.

To utilize the auxiliary knowledge, such as the rating data from other domains, [23] considers collaborative matrix factorization (CMF). Let R_s denote the rating matrix from another domain. Suppose two domains share the same users. It aims to minimize the following objective function:

$$\min_{U,V,W} \alpha \parallel (R - UV^T) \odot I \parallel_f^2 + (1-\alpha) \parallel (R_s - UW^T) \odot I \parallel_f^2 + \mathfrak{R}(U, V, W)$$

where W is the latent matrices for items in source domain, $\mathfrak{R}(U, V, W)$ is the regularization term, and α is the parameter to adjust the relative effects of two domains. Some recently proposed cross-domain MF methods [13, 1] have similar objectives, but employ different regularization terms. Obviously, these methods transfer knowledge globally; as for each rating in R, the same regularization $(R_s - UW^T)^2$ is imposed.

The methods proposed in [14, 9] incorporate relationship information between users, in order to make some users influence each other more than those remaining. However, they'd require additional information, such as social network and user properties, to build such regularization. This type of information may not always be available, such as Netflix[2]. In addition, these approaches do not consider the similarity between items. One advantage of the proposed method is that it does not require such additional knowledge. Another observation is that these previously proposed methods still decompose the rating matrix uniformly, and could easily run into over-fitting problem unless parameters are carefully tuned.

Several other CF approaches also exploit hierarchical approach. For example, to accelerate convergence, [7] proposes a multilevel method for nonnegative matrix factorization. However, this method is very different from the proposed approach in this paper, since it still does not consider the different contexts and the over-fitting problems. Recently, [17] introduces a hierarchical method for response prediction. The major difference is that we build the hierarchical structure automatically from training data, while the approach in [17] requires a manually fixed structure, which is a human-cost job.

A few theoretical analysis have been proposed. [24] proposes generalization bounds for low-rank matrices approximation. [25] suggests three different criteria: rank, trace-norm and max-norm to measure the complexity of low-rank matrices approximation. [21] analyzes the error bounds on collaborative filtering through PAC-Bayesian analysis. Recently, [22] proposes a general bound which does not use the uniform assumption.

2.2 Random Partition on Tree Structure We adopt the idea of random partition from Random Decision Trees[3] [4, 3] which is applicable for classification and regression to partition the rating matrix and build ensemble. To construct one tree in the ensemble, the feature at a non-leaf node is chosen randomly from the "remaining" features. Each time the feature is chosen, a random threshold is selected. According to the selected feature and threshold, the instances are partitioned into two parts according to their values on the selected feature. A tree stops growing any deeper if one of the following conditions is met: (1) a node becomes empty or there are no more examples to split in the current node or (2) the depth of tree exceeds some limits. For classification, each node of the tree records class distributions,

[2] http://www.netflixprize.com/

[3] Open source project for Random Decision Tree is available from http://www.dice4dm.com

Algorithm 1 RPMF

Input: Rating Matrix: R, Number of Latent Factors: d, Number of Trees: N, Depth of Trees: h
Output: Tree Ensemble with Latent Matrices at Each Node
for $i = 1$ to N **do**
  Build Tree T_i
  Decompose R into U and V using Eq.(3.4)
  Maintain U and V at the current node
  Select a latent factor from U, V and a splitting point randomly
  Partition R into R_1 and R_2 using the selected factor
  for $j \in \{1, 2\}$ **do**
    IF $|R_j|$ is small or the depth is exceed h
      Let its corresponding node as leaf node
    ELSE
      Decompose R_j recursively
  end for
end for
Return $\{T_i\}_{i=1}^{N}$

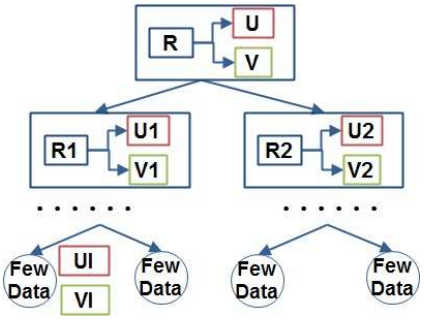


Figure 1: Main Flow

and each tree outputs a class probability distribution during prediction. The class distribution outputs from multiple trees are averaged as the final output. We note that, the advantage of random partition is that it does not require supervised information to select either features or thresholds. Thus, the proposed algorithm can apply the idea to select a latent factor from low-rank matrices without labels and then split the rating matrix accordingly.

3 Hierarchical Matrix Factorization based on Random Partition

We describe the proposed RPMF algorithm, explain it using gradient boosting formulation and discuss its complexity.

3.1 Algorithm Description The main flow can be found in Algorithm 1 and Figure 1. The motivation is to group the ratings generated by the similar users and items into the same node using random partition.

For each tree, we perform random sampling on the original rating matrix. Then, at each node, we factorize the maintained rating matrix R using basic matrix factorization.

$$\min_{U,V} \parallel (R - UV^T) \odot I \parallel_f^2 \tag{3.3}$$

where U is a $m \times d$ matrix, V is a $n \times d$ matrix, m is the number of users, n is the number of items, d is the number of factors, and $\parallel * \parallel_f$ is the Frobenius norm. After decomposition, matrix $U = \{U_1, \ldots, U_m\}, U_u = \{u_1, \ldots, u_d\}$ is a latent representation of users and V is the corresponding representations of items. Then, the users and items can be represented by latent factors, $\{u_1, \ldots, u_d\}$ and $\{v_1, \ldots, v_d\}$, as shown in Figure 2. The user factors indicate users' interest distribution on some latent topics, while the item factors present items' membership to these topics. To partition the rating matrix R, we select a latent factor and a splitting value randomly, such as the second column of U in Figure 2(b). In this example, suppose we pick a random threshold of 0.4. After that, the current rating matrix is divided into two parts according to the value of this latent factor, such as the example in Figure 2(c). In this example, the rating matrix is split between the second and third rows according to the second latent factor of U and the chosen threshold. As the latent values of the first and second users are similar, their ratings are partitioned into the same node, but the ratings from the third and fourth users are partitioned into a different node. This process is repeated recursively until either the number of ratings in the node is too small or the depth of the tree exceeds a given threshold. Thus, at each node we maintain two latent matrices U and V to construct the rating interactions between users and items. We set the latent matrices of parent nodes as the initial solutions for the child nodes. For the non-root nodes, it decomposes the rating matrix at the node as

$$\begin{aligned} \min_{U,V} \quad & \parallel (R - UV^T) \odot I \parallel_f^2 \\ + \quad & \nu \cdot \Big(\parallel U - U_p \parallel_f + \parallel V - V_p \parallel_f \Big) \end{aligned} \tag{3.4}$$

where U_p and V_p are the latent matrices in parent node, ν is a trade-off parameter, and $\parallel * \parallel_f$ is the Frobenius norm. We also perform line-search on ν to form a additive model. During prediction, for a given user-item pair, we let it go through from the root to the leaf node on each tree. At each node we obtain a partial prediction $\tilde{R}_{ij}^k = U_i^k V_j^k$. At the end, the predictions from all trees are combined to obtain the final prediction.

$$\tilde{R}_{uv} = \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{h} \frac{1 - \nu_i}{\sum_i (1 - \nu_i)} U_u^{ij} V_v^{ij} \tag{3.5}$$

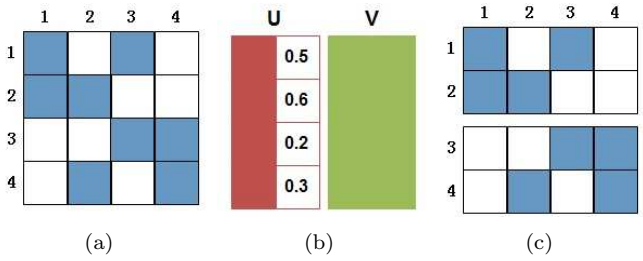


Figure 2: Illustration for Division

By incorporating random partition to obtain local matrix factorization, the proposed method RPMF has the following properties: (1) ratings with similar users and items appear on common path and then impose more influence on each other; (2) the prediction of a user-item pair is combined with different latent matrices, generated from different contexts; (3) model-averaging is performed to avoid over-fitting without parameter tuning.

3.2 Model-averaging Gradient Boosting Explanation Besides the properties described above, contextual and none over-fitting, the proposed method is a model-averaging gradient boosting model. Specifically, each tree is a gradient boosting model and the tree ensemble is a model-averaging model. Let $f(U, V) = UV^T$ denote the single model at each node and $F(U, V)$ be the final model obtained along a decision path. Recall Eq.(3.4), we have

$$F(U, V) = \sum_{p=1}^{h} \frac{1 - \nu_p}{\sum_p (1 - \nu_p)} f_p(U, V) \tag{3.6}$$

We start with an initial model $f_1(U, V)$ at the root node. After that, we approximate the final model using the sub-rating matrix:

$$\min_{U,V} \parallel (\tilde{R}_p - UV^T) \odot I_p \parallel_f^2 \tag{3.7}$$

where I_p is the current index matrix and $\tilde{R}_p$ is the current "pseudo"-residuals

$$\tilde{R}_{pij} = -\frac{\partial \mathcal{L}(R_{pij}, F_{ij})}{\partial F_{ij}}\Big|_{F=F_{p-1}} \tag{3.8}$$

where $\mathcal{L}$ is the loss function, e.g, the squared loss used in this paper, and F_{ij} is the prediction on $\tilde{R}_{pij}$. Based on this, we update the model in a forward "stage-wise" manner.

$$F_p(U, V) = F_{p-1}(U, V) + (1 - \nu_p) f_p(U, V) \tag{3.9}$$

In addition, since we perform random sampling for each tree and average the predictions, we effectively generate a model-averaging model. As gradient boosting method

can reduce bias [6] and model-averaging can decrease variance [19], the proposed method has good generalizability as analyzed in Section 4.2.

3.3 Complexity Analysis When SGD is used to solve the objective of matrix factorization, it requires N_I iterations to guarantee convergence. At each iteration, it exhausts all ratings in R and updates the latent matrices with dimension d. The complexity is thus $O(N_I * |R| * d)$. For SVD++, it updates the constraints Y from implicit feedbacks. For a rating R_{ij}, it takes $|R_i| * d$ to update, where $|R_i|$ is the number of ratings provided by user u. Thus its time complexity is $O(N_I * d * \sum_i |R_i|^2)$. For BPMF, it needs to compute the inverse of a $d \times d$ matrix, which takes $O(d^3)$. Its total complexity is $O(N_I * |R| * d^3)$. However, for RPMF, one needs to decompose the sub-rating matrices at each node and builds N trees. Thus, its overall complexity is $O(N_I^{'} * |R| * d * h * N)$, where h is the height of trees, and $N_I^{'} \leq N_I$ since the decomposition process converges quickly towards leaf node. In practice, $h = 3$ and $N = 5$ are enough to make the proposed algorithm achieve good performance. Instead, the typical setting of d is larger than 10 and then RPMF is faster than both SVD++ and BPMF.

3.4 Differences between RPMF and Random Partition in Classification/Regression There are three main differences between the Random Partition for classification/regression (Random Decision Tree [4]) and RPMF: (1) RPMF decomposes rating matrix in order to obtain latent features while the classification method only uses existing features. (2) RPMF combines the predictions at every single node (non-leaf and leaf) on the decision path from root to leaves, while classification method uses class distributions at leaf nodes. (3) RPMF exploits a gradient boosting process when splitting node, while the classification method does not.

4 Formal Analysis

We analyze the proposed algorithms from three perspectives: (1) the error of the low-rank matrix approximation at each node can be bounded; (2) a single tree in RPMF is an additive as well as a boosting model, which guarantees good generalizability by reducing the bias; (3) tree ensemble is model-averaging and further improves the prediction performance by reducing the variance.

4.1 Error Bound for Low-Rank Approximation Now we propose a new error bound for low-rank matrix approximation for multi-class ratings. It is important to note that the bound in [24] is derived only for two class rating, i.e, 0 and 1. Let R^* be the ideal rating matrix to approximate, R be the observed rating matrix, $\mathcal{L}(R^*, UV^T) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}|R^*_{ij} - U_iV_j|$ and $\mathcal{L}(R, UV^T)$ be the corresponding empirical error. Let $o = \{o_1, \ldots, o_d\}$ be the singular values of R, and $\Sigma = diag(o)$ be a diagonal matrix where the ith element on the diagonal is o_i. We minimize the following objective with respect to U and V.

$$\| (R - U\Sigma V^T) \odot I \|_f = \sum_{R_{ij}\in R} |R_{ij} - \sum_{k=1}^{d} o_k U_{ik} V_{jk}|$$

We then derive a bound on empirical error as follows:

THEOREM 4.1.

$$\mathcal{L}(R, UV^T) \leq \frac{1}{mn}\sum_{R_{ij}\in R}\left(\sum_{k=1}^{d}|1-o_k|U_{ik}V_{jk}^T + \sqrt{\sum_{k=d+1}^{rank(R)} o_k^2}\right)$$

Proof. This is straight-forward. Let UV^T subtract $U\Sigma V^T$, we obtain

$$\begin{aligned} \| UV^T - U\Sigma V^T \|_f &= \| U\Omega V^T - U\Sigma V^T \|_f \\ &= \sum_{R_{ij}\in R}(|1 - o_k|U_{ik}V_{jk}^T) \end{aligned}$$

where Ω is an identity matrix. Then by incorporating the theorem in [8], we can easily derive the theorem.

Now we develop the generalization bound, a tradeoff between the empirical error and the model complexity.

THEOREM 4.2. *For any matrix $R^* \in \{1, \ldots, Q\}^{m\times n}$, $n, m > 2$, $\delta > 0$ and integer $k < rank(R)$, with probability of at least $1 - \delta$ over choosing a training set R of entries in R^* uniformly among all subsets of $|R|$ entries, the generalization error of approximating R^* satisfies:*

$$\begin{aligned} (4.10)\qquad & \mathcal{L}(R^*, UV^T) < \mathcal{L}(R, UV^T) \\ & +(Q-1)|R|\sqrt{\exp\left(\frac{\log Qd(n+m)\log(\frac{16em}{d}) - \log\sigma}{|R|}\right)} \end{aligned}$$

Proof. First, we assume the ratings in R are selected independently and uniformly from R^*. Then $|R_{ij} - U_iV_j^T|$ is a random variable with probability defined in $|R^* - UV^T|$. Furthermore, the mean of $|R|\mathcal{L}(R, UV^T)$ is $|R|\mathcal{L}(R^*, UV^T)$ and the standard variance $\sigma \leq Q - 1$. By applying Chebyshev's inequality, for $\forall a > 0$, we have

$$(4.11)\quad Pr\Big[\mathcal{L}(R^*, UV^T) \geq \mathcal{L}(R, UV^T) + a|R|\sigma\Big] \leq \frac{1}{a^2}$$

We should propose a reasonable a for a tight bound. Note that $|R^*_{ij} - U_iV_j|$ depends on the complexity of the

model: d, the rank of both U and V. We consider the number of patterns that can be generated by UV^T.

$$M(m,n,d) = \left\{ UV^T \in \{1,\ldots,Q\}^{m\times n} \right\} \tag{4.12}$$

Let $N(m,n,d) = |M(m,n,d)|$ denote the number of such patterns. By setting $a = \sqrt{\exp(\frac{\log N(m,n,d) - \log \sigma}{|R|})}$, we obtain

$$Pr\Big[\mathcal{L}(R^*, UV^T) \geq \mathcal{L}(R, UV^T) + (Q-1)|R|\sqrt{\exp(\tfrac{\log N(m,n,d) - \log \sigma}{|R|})}\Big] \leq \sigma \tag{4.13}$$

The rest of the proof relies on the size of $N(m,n,d)$. We follow the main idea from [24]. Let us consider the $d(m+n)$ entities of U and V as variables, and the mn entities of UV^T as polynomials of degree two over these variables: $\hat{R}_{ij} = \sum_{k=1}^{d} U_{ik}V_{jk}$. We have the following Lemma:

LEMMA 4.1. *If the $R \in \{0,1\}^{m\times n}$, $N(n,m,d)$ has an upper bound as follows.*

$$N(m,n,d) \leq \Big(\frac{16emn}{d(m+n)}\Big)^{d(m+n)} \leq \Big(\frac{16em}{d}\Big)^{d(n+m)}$$

For proof of this Lemma, please refer to the one in [24]. Let us further consider a more general setting, for each rating in $\{1,\ldots,Q\}$, it can be represented by binary code with length $\log Q$. Then this size $N(m,n,d)$ can be re-written as

$$N(m,n,d) \leq (\frac{16em}{d})^{d(n+m)\log Q} \tag{4.14}$$

By substituting this term into Eq.(4.13) and rearranging other terms, we establish Theorem 2.

4.2 On Additive and Boosting Properties We analyze why the performance can be boosted by decomposing the rating matrix locally. As we apply the latent matrices obtained by the parent nodes as the "seeds" for the child nodes (See Eq.(3.4)), the learning at the child nodes can be considered as an approximation to the residual from the ancestor nodes. Let U^p and V^p be the latent matrices obtained from the root to the parent of the current node to approximate the residuals, and h' be the height of the current node. We have

$$\min_{U^*,V^*} \| \big((R - \sum_{p=1}^{h'-1}(1-\nu_p)U^pV^{pT}) - U^*V^{*T}\big) \odot I \|_f^2$$

where ν_p is the regularization parameter for the decomposition at each node. Thus, the model obtained at the leaf nodes can be represented as

$$F(U,V) = \sum_{p=1}^{h}(\frac{1-\nu_p}{\sum_p(1-\nu_p)})U^pV^{pT} \tag{4.15}$$

This is a generalized additive model as well as a boosting model. We analyze from two different perspectives, one is from regression graph [10] and the other is a geometric bound. For convenience of analysis, we assume the height of the trees is the same as the number of latent factors. We show that the error of Eq.(4.15) is related to the error of a generalized additive model, and with high probability, every regression graph (of every size) has empirical error close to its generalization error. Let $G(U,V) = f(\sum_{p=1}^{h} g^p)$ be a generalized additive model, where $f : \mathcal{R} \to [1,\ldots,Q]$. Then we have the following theorem.

THEOREM 4.3. *Let the differential of f be $|f'(a)| \leq \alpha$ and the one of g be $|g'(b)| \leq \beta$ for all $a \in R$ and $b \in [1,\ldots,Q]$. For any $\delta > 0$, with probability $1-\delta$ over training data R of $|R|$ i.i.d samples of R^*, the relation between the error of G and the one in Eq.(4.15) is as follows*

$$\mathcal{L}(R^*, G) = \mathcal{L}(R, G) + \frac{3(1+\alpha\beta d)\ln(5|R|d/\delta)}{|R|^{1/7}} \tag{4.16}$$

In our context, $g^p = U^pV^{pT}$ and $G = F$. Then we have a uniform convergence:

THEOREM 4.4. *For any $\delta > 0$,*

$$Pr\Big[|\mathcal{L}(R,F) - \mathcal{L}(R^*,F)| > 3.8\sqrt{\tfrac{\log Qd(m+n)\ln 3|R|d^2(m+n)/\delta}{|R|}}\Big] \leq \delta \tag{4.17}$$

For proofs of these theorems, one can refer to [10]. Obviously, this error bound linearly increases with d while the error bound in Eq.(4.10) exponentially increases with d. It means that the proposed algorithm is less sensitive to the dimension d.

In addition, for the boosting property, we analyze each single tree based on margin theory for boosting algorithms. As follows, we first analyze the training error of a single model and then show that the generalization error can be bounded. Let ϵ_p be the empirical error of the p-th model in one path, where

$$\epsilon_p = \sum_{r_{ij}\in R} P_{ij} \cdot [|g_{ij}^p - r_{ij}| < b] \tag{4.18}$$

where b is constant to transform the loss in regression prediction into 0/1 loss, $P_{ij} = 1$ if r_{ij} is in the current sub-rating matrix and $P_{ij} = 0$ otherwise. This

formulation can be considered as the weights assigned to the rating by the algorithm. The predicate $[\cdot] = 1$ if $\cdot$ is true and $[\cdot] = -1$ otherwise. In addition, we re-define the loss as $\mathcal{L}(R^*, F) = \sum_{r \in R^*} \big(2*(F(r)-r)/(Q-1)-1\big)$ and then the theoretical results of boosting can be applied. Let $\gamma_p = 1/2 - \epsilon_p$, which measures the advantage of the single model at each node over a trivial model. Now, we consider a different error: margin-error, $Pr[\mathcal{L}(R^*, F) < \theta]$, which can be proven to be small. Let

$$\bar{\gamma_\theta} = \frac{\theta}{2} + \frac{1}{\sqrt{2}}\sqrt{\log 2 - \Psi(\frac{1-\theta}{2})} \tag{4.19}$$

where $\Psi(u) = -u \log u - (1-u)\log(1-u), 0 \le u \le 1$, is the binary entropy function. Then we obtain

THEOREM 4.5. *Let $\epsilon_p = \bar{\epsilon_\theta} + \delta_h$. Then the empirical margin-error of the composite model obtained at step h is bounded from the above by*

$$Pr[\mathcal{L}(R, F) < \theta] \le \exp\Big\{ -2(2\bar{\epsilon_\theta} - \theta)\sum_{p=1}^{h}\delta_h - 2\sum_{p=1}^{h}\delta_h^2 \Big\}$$

This error is bounded by a finite value smaller than 1 if

$$\lim_{h\to\infty} \inf \sum_{p=1}^{h} \epsilon_p \ge c > 0 \tag{4.20}$$

and converges to 0 if either $c = \infty$, or if c is finite and $\sum_{p=1}^{h} \epsilon_p^2$ converges to infinity.

The proof can be found in [16]. Based on this analysis, we can derive a new generalization bound [5]. Let $\mathcal{H}$ be a class of model of VC-dimension $d_{\mathcal{H}}$, and denote by $co(\mathcal{H})$ the convex hull of $\mathcal{H}$, namely, for $\alpha_p \ge 0$, and $\sum_p \alpha_p = 1$

$$co(\mathcal{H}) = \Big\{ F : F(U,V) = \sum_p \alpha_p g_p(U,V) \Big\}$$

As such, we obtain

THEOREM 4.6. *With probability at least $1-\delta$, for every $F \in co\mathcal{H}$ and $\theta > 0$, the generalization error*

$$\begin{aligned} Pr[\mathcal{L}(R^*, F) \le 0] &\le Pr[\mathcal{L}(R, F) < \theta] \\ &+ O(\frac{1}{\theta}\sqrt{\frac{d_{\mathcal{H}}}{|R|}}) + O(\sqrt{\frac{\log\frac{1}{\delta}}{|R|}}) \end{aligned}$$

This bound is based on the margin and VC-dimension while the VC-dimension is related to the number of latent features. In addition, the bound increases sub-linear to the VC-dimension, which suggests that the proposed single model with one single tree can achieve a better generalizability than the basic factorization model. Although the analysis is for binary rating, it can be extended to multi-class using the same transform in Eq.(4.14).

4.3 Generalization of Model Averaging We analyze the reason to combine multiple trees to generate ensemble in the algorithm: (1) the expected error of the averaging ensemble does not exceed the one of a single model; (2) the ensemble has a tight bound. Let $F_E = \mathbf{E}_{F\sim M(m,n,d)} F(U,V)$ denote the averaging ensemble and $d(F_E, F^*)$ denote the difference between the error of F_E and the ideal hypothesis F^*. We have the following lemma.

LEMMA 4.2. *For any single model F and a model-averaging ensemble F_E, $d(F, F^*) \ge d(F_E, F^*)$.*

Proof. The difference $d(F, F^*)$ is:

$$\begin{aligned} d(F, F^*) &= \mathbf{E}_{F\sim M(m,n,k)}\big(F(U,V) - F^*(U,V)\big)^2 \\ &= \mathbf{E}_{F\sim M(m,n,k)}\big(F(U,V)^2 \\ &- 2F(U,V)F^*(U,V) + F^*(U,V)^2\big) \end{aligned}$$

On the other hand, the difference $d(F_E, F^*)$ is

$$\begin{aligned} & d(F_E, F^*) \\ &= \mathbf{E}\big(\mathbf{E}_{F\sim M(m,n,k)}F(U,V) - F^*(U,V)\big)^2 \\ &= \mathbf{E}\Big(\big(\mathbf{E}_{F\sim M(m,n,k)}F(U,V)\big)^2 \\ &- 2\big(\mathbf{E}_{F\sim M(m,n,k)}F(U,V)\big)F^*(U,V) + F^*(U,V)^2\Big) \\ &\le d(F, F^*) \qquad as \qquad \mathbf{E}[F]^2 \le \mathbf{E}[F^2] \end{aligned} \tag{4.21}$$

Thus, the claim is held true.

Further more, since the averaging ensemble can reduce the variance, lower bound can be achieved. When we train N independent models, the prediction variance reduce to σ/N while the mean keeps the same. Thus, by recalling Eq.(4.10), we obtain a lower bound:

$$\begin{aligned} &\mathcal{L}(R^*, F_E(U,V)) < \mathcal{L}(R, F_E(U,V)) \\ &+ \frac{(Q-1)|R|}{N}\sqrt{\exp\Big(\frac{\log Qd(n+m)\log(\frac{16em}{d}) - \log\sigma}{|R|}\Big)} \end{aligned}$$

The proof is similar to the one of Theorem 2 and it is omitted here. We introduce a tighter bound from [18] where the complexity of the model depends on the VC-dimension. Since a single tree can be considered as a linear combination of d rank-1 matrices [24] and the ensemble averages all predictions from every trees, its VC-dimension is Nd and the bound is

THEOREM 4.7. *Let $\tilde{M}$ be the family of models formed by linearly combining finitely many single models from M. Assume that the family $\tilde{M}$ has finite VC-dimension $v = Nd$. Then there exists $\delta_0 > 0$ such that for every $0 < \delta \le \min(\delta_0, 1)$, with probability at least $1 - \delta$*

$$\mathcal{L}(R^*, F_E(U,V)) \le \mathcal{L}(R, F_E(U,V)) + \epsilon(\delta, |R|) \tag{4.22}$$

Table 2: Summary of Datasets

Name	#User	#Item	$\|R\|$	Range
Movielens-100K	943	1682	$100K$	$\{1\sim5\}$
Movielens-1M	6040	3952	$1M$	$\{1\sim5\}$
Movielens-10M	71567	10681	$10M$	$\{1\sim5\}$
Eachmovie	74424	1648	$2.8M$	$\{1\sim6\}$
Netflix	480189	17770	$100M$	$\{1\sim5\}$
Yahoo! Music	$\sim 1M$	624961	$250M$	$\{0\sim100\}$

$$\epsilon(\delta,|R|) = \frac{2v-3}{4C_0\sqrt{|R|}}\Big(1+\big(1+0.5(\tfrac{4C_0}{2v-3})^2 \big[\log\tfrac{1}{\delta}+(v-1)\log\tfrac{KC_0^2}{(v-1)e^2}+\log\tfrac{Ke}{C_0}\big]\big)^{0.5}\Big),$$

$$\delta_0 = \frac{K}{C_0}\Big(\frac{KC_0^2}{v-1}\Big)^{(v-1)}e^{-2C_0^2}$$

K and C_0 are universal constants. In particular, $\epsilon(\delta,|R|) \geq C_0/\sqrt{|R|}$ for all $0<\delta\leq\min(\delta_0,1)$.

This theorem indicates that if we can improve the training approximation while keeping the model complexity small, a better generalization can be achieved. We also observe the bound is increase with the d linearly which improves the one using basic factorization model.

5 Experiments

We evaluate the proposed approach RPMF on several movie and music collaborative filtering datasets, including the MovieLens-1M, EachMovie, MovieLens-10M[4], Netflix[5] and Yahoo! Music Recommendation[6] and compare with four baselines: basic Matrix Factorization, SVD++ [11], Bayesian Probabilistic Matrix Factorization (BPMF) [20] and Gaussian Process Latent Variable Model (GP-LVM) with RBF kernel [12]. To test RPMF's sensitivity to parameters: number of trees and height of trees, we conduct extensive experiments on MovieLens-100K dataset. The summary of datasets can be found in Table 2. We set the number of trees in RPMF as $N=5$ and the height as $h=3$. We evaluate the results in 5-CV. The prediction error is measured in RMSE.

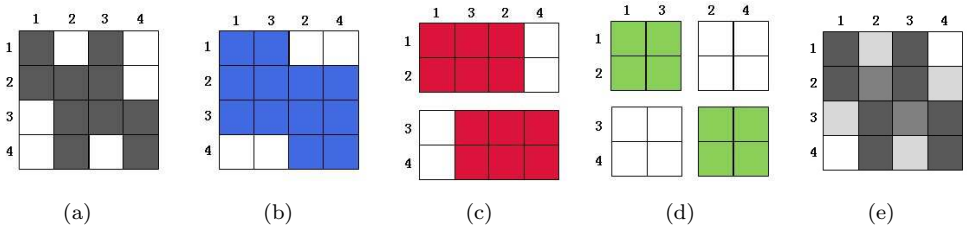
$$RMSE = \sqrt{\sum_{R_{ui}\in T_E}(R_{ui}-\hat{R}_{ui})^2/|T_E|} \tag{5.23}$$

where T_E is the testing set and $\hat{R}_{ui}$ is the prediction.

5.1 Synthetic Example To illustrate the process of RPMF, we introduce one synthetic example as shown in Figure 3 (a) using 4 users, 4 items and ratings in $\{0,1\}$, where the black represents 1 and white represents 0. We set the dimension of latent matrices as 1 and use only one tree with height 3. Figure 3 (b)$\sim$(d) denote the approximations at each level. We observe that at the root, ratings R_{13} and R_{24} receive wrong approximations. Then, based on the similarity between users, the matrix is divided into two parts between users 2 and 3. We execute this process recursively and obtain the results as shown in Figure 3 (c) and (d). Obviously, both of them have two wrong approximations. However, after combination of predictions at each level, wrong approximations can be filtered by setting a simple threshold and we obtain a correct result as shown in Figure 3 (e). This implies that, by considering the different contexts at each partitions, higher accuracy can be obtained.

5.2 Performance on Small Datasets Table 3 summarizes the RMSE of MF, SVD++, BPMF, GP-LVM and the proposed RPMF on the two databases. For the MovieLens-1M collection, as highlighted in bold, RPMF consistently outperforms GP-LVM, SVD++, BPMF and MF on all different latent dimensions. When the dimension is 30, we notice that the RMSE of RPMF is at least 0.12 less than MF. If we overlook the model differences, compared to its competing methods, RPMF on average achieves 0.09, 0.04, 0.01 and 0.02 less RMSE as compared to MF, SVD++, BPMF and GP-LVM respectively. The better performance of RPMF over SVD++ can be ascribed to both the hierarchical matrix factorization and ensemble prediction implemented in RPMF, as it utilizes the predictions from different contexts and avoid over-fitting. In addition, by utilizing hierarchical structure, ratings from similar items will be partitioned unto the same nodes and then these similar items can impose higher impacts on each other. Although SVD++ explores implicit feedbacks to improve its accuracy, its prediction is still uniform without considering contexts, i.e., users' different mood and affections at different times. This also explains the advantage of RPMF over BPMF and GP-LVM, where BPMF is proposed to reduce over-fitting and GP-LVM utilizes the non-linearbility, but neither of them explores contextual information. Moveover, in all cases, RPMF outperforms MF on EachMovie datasets. For comparison to other three baselines, RPMF beats SVD++ consistently, fails to defeat BPMF when $d=5$ but winning all other 5 pairwise studies and performs as well as GP-LVM. This, from the empirical perspective, provides justification to the theoretical analysis of model-averaging gradient boosting model in Section 4, where RPMF achieves a lower bound.

[4] http://www.grouplens.org/
[5] http://www.netflixprize.com/
[6] http://kddcup.yahoo.com/



(a) (b) (c) (d) (e)

Figure 3: Synthetic Example of RPMF's Process

Table 3: RMSE in MovieLens-1M and EachMovie

Methods	MF	SVD++	BPMF	GP-LVM	RPMF	MF	SVD++	BPMF	GP-LVM	RPMF
Dimensions	MovieLens-1M					EachMovie				
d=5	0.9125	0.8723	0.8735	0.8873	**0.8614**	1.1832	1.1342	1.1227	**1.1221**	1.1271
d=10	0.9192	0.8785	0.8690	0.8829	**0.8549**	1.1953	1.1523	1.1210	1.1171	**1.1153**
d=15	0.9384	0.8892	0.8671	0.8837	**0.8550**	1.2196	1.1722	1.1203	1.1163	**1.1150**
d=20	0.9563	0.8992	0.8692	0.8867	**0.8592**	1.2472	1.1899	1.1173	1.1143	**1.1157**
d=25	0.9730	0.9134	0.8702	0.8871	**0.8618**	1.2706	1.2071	1.1208	**1.1178**	1.1189
d=30	0.9908	0.9184	0.8724	0.8889	**0.8672**	1.2921	1.2201	1.1285	**1.1198**	1.1215

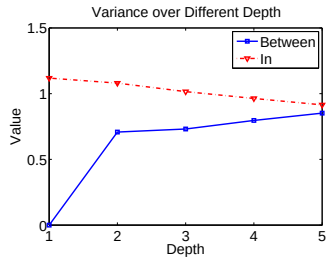
5.3 Performance on Large Datasets Table 4 summarizes the RMSE for the different baselines as well as the proposed approach on two large datasets: MovieLens-10M and Netflix. Using GP-LVM requires a kernel, which is computationally infeasible for the Netflix dataset, due its large size, we omit its results. It is obvious to note that the proposed approach, with different number of latent factors, significantly outperforms the basic MF, SVD++ and GP-LVM, and performs as good as BPMF. Importantly, the proposed approach uses much smaller latent dimensions than the others which infers a faster computation. In the proposed approach, larger dimensional spaces resulted in better performance. The reason is that the latent space is trained using more data, and thus is less prone to over-fitting, as analyzed in Section 4. For the Netflix dataet, the proposed approach performs as well as SVD++, one of the most competitive approach with carefully parameter tuning on the leaderboard. On the other hand, since RPMF considers different contexts, it outperforms other approaches. Similarly, the main reason is that the latent space required by RPMF is small. Its performance with only 5 dimensions defeats BPMF using 15 dimensions. Overall, RPMF performs well, and its very competitive and efficient for large datasets. The results on high dimensional problems (d=200) are presented in Figure 4(f). The performances of RPMF and SVD++ are similar and slightly better than BPMF. In addition, RPMF improves MF significantly by 0.037 on RMSE. These imply that it is suitable for real-world applications.

In addition, we evaluate the RPMF method using the Yahoo! Music Recommendation dataset. This dataset is more difficult than the other two, due to the following two reasons: (1). Its rating scope is wider, from 0 to 100; (2). Its size is much bigger than others. These make other approaches fail. SVD++ and BPMF fail to give a result in a feasible time scope. The results of MF and RPMF are 27.63 and 24.95 respectively. We notice that the result we obtain here is less than the top ones in the leader board. According to the official report [2] and the results on the leader board [7], the result of the RPMF is at 100-th out of 1200 reported entries or approximately top 8.3%. From the official report, these leading results not only exploit the rating information but also other related knowledge, such as music taxonomy, not used in the proposed approach. In addition, these methods perform model blending by combining hundreds to even thousands of different algorithms, while the proposed approach uses a single algorithm by fixing parameters $h = 3$ and $N = 5$ without any tuning.

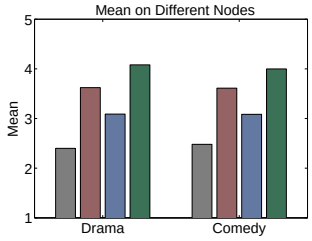
5.4 Performance Analysis We analyze three issues in detail to give some insights to understand the performance of the proposed algorithm: (1) How does RPMF improve the performance when the ratings are generated from different contexts, e.g, different types of items? (2) How does RPMF avoid over-fitting? (3) How is the efficiency of RPMF as compared to state-of-the-art MF methods?

To evaluate the performance of RPMF over different contexts, we design three experiments using the "MovieLens-100K" dataset. To do so, we compare the rating difference between tree nodes against those

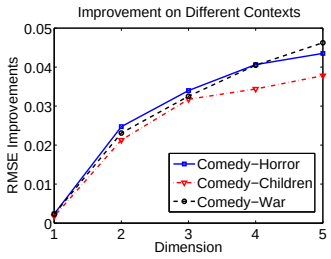
[7] http://kddcup.yahoo.com/leaderboard.php?track=1&n=100



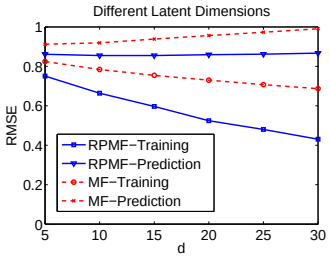
(a) Contexts on Depths



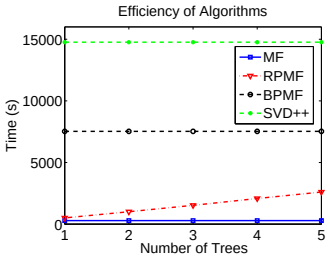
(b) Contexts on Nodes



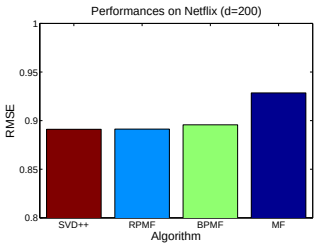
(c) Improvements on Different Types



(d) Over-fitting

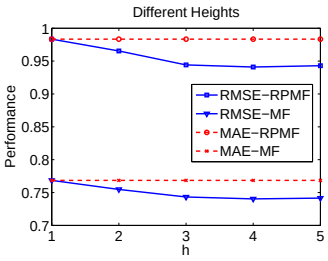


(e) Efficiency

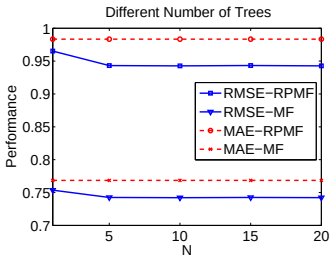


(f) RMSE in Netflix

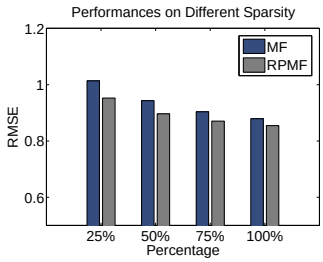
Figure 4: Performance Analysis



(a) Height



(b) Trees



(c) Sparsity

Figure 5: Parameter Analysis

within tree nodes at each level in a tree of RPMF. As shown in Figure 4(a), the variance of ratings between nodes increases as the tree grows deeper, while the variance within nodes decreases. This implies that, using the random partition approach, similar ratings are grouped together and thus can give higher impact on each other, but dissimilar ones are partitioned into different nodes. These strategies lead to context-aware decomposition. A second example is given in Figure 4(b). We extract ratings of two largest movie categories, drama and comedy, and let them pass to the third level in the tree. Then they are grouped into 4 or $2^{(3-1)}$ nodes. Obviously, the means of ratings in different nodes are quite different. This shows that they are generated under different contexts. In addition, we extract movies from 4 genres with different ratings value: Comedy (3.39), Children (3.35), Horror (3.29) and War (3.82). The difference in rating values means that they are generated under different contexts. Then we generate 3 datasets: Comedy-Children, Comedy-Horror and Comedy-War. From Figure 4(c), we observe that as differences among ratings gets larger, RPMF can achieve higher improvements. The obvious reason is that RPMF can capture the difference in contexts using rating information and group similar items into same nodes. To examine how well RPMF avoids over-fitting, we plot the training and testing errors of MF and RPMF on "MovieLens-1M" in Figure 4(d). The observation is that RPMF can easily avoid over-fitting; it is insensitive to the dimension of the latent matrices while MF is very sensitive. As shown in the figure, when RPMF achieves lower training error with larger latent dimensions, it still maintains similar prediction error on testing data. On the other hands, the testing error of MF increases significantly. For efficiency comparisons, the running time of each algorithm is plotted in Figure 4(e). As the number of trees increases, the computational cost of RPMF increases linearly and slowly. As compared with SVD++ and BPMF, RPMF is much faster (the training cost is insignificant) and this confirms the complexity analysis in Section 3.4.

Table 4: RMSE in MovieLens-10M and Netflix

Methods	MF	SVD++	BPMF	GP-LVM	RPMF	MF	SVD++	BPMF	GP-LVM	RPMF
Dimensions	MovieLens-10M					Netflix				
d=5	0.9115	0.8762	**0.8545**	0.8769	0.8564	0.9589	**0.9189**	0.9210	-	0.9192
d=10	0.9201	0.8745	**0.8472**	0.8740	0.8499	0.9575	**0.9142**	0.9153	-	0.9144
d=15	0.9454	0.8812	0.8567	0.8777	**0.8483**	0.9590	0.9101	0.9117	-	**0.9087**
d=20	0.9637	0.8892	0.8589	0.8807	**0.8501**	0.9665	0.9067	0.9085	-	**0.9054**
d=25	0.9741	0.9004	0.8601	0.8851	**0.8568**	0.9770	**0.9012**	0.9059	-	0.9022
d=30	0.9899	0.9093	0.8634	0.8909	**0.8592**	0.9801	**0.8979**	0.9044	-	0.8983

5.5 Parameter Analysis As shown in Section 3, two parameters N and h need to be set before running RPMF. We argue that they do not need too much tuning in the Introduction, and now we conduct an extensive experiment on MovieLens-100K dataset to show their effects. To look deeper into the algorithm, we also plot the performance of MF and introduce one more criterion: MAE.

$$\text{MAE} = \sum_{R_{ui} \in T_E} |R_{ui} - \hat{R}_{ui}| / |T_E| \tag{5.24}$$

We set N as 1, 5, 10, 15 and 20, and increase h from 1 to 5 with step length 1, thus to obtain the results of RPMF in Figure 5(a) and (b). We observe that, as height increases, the RMSE and MAE drop down and then converge quickly. When h is 1, the model just consider one context and cannot predict precisely. As the height h is equal to or bigger than 3, both RMSE and MAE reach the least in value. As the number of trees N increases, the prediction error decreases. But when $N \geq 5$, both RMSE and MAE do not change significantly. Thus, as stated in Section 3, small N and h are sufficient to improve the prediction accuracy, while maintaining high efficiency. Moreover, for every setting in this study, RPMF outperforms MF. To test the sparsity issue, we sample 20,000 users with their ratings from Netflix dataset randomly and sample again to generate four datasets with different sparsity. As shown in Figure 5(c), under different sparsity, RPMF outperforms MF consistently. In addition, RPMF can achieve the performance of MF with significantly fewer ratings. For example, with 75% training data, RPMF performs comparably against MF.

6 Conclusion

Though as one of the most popular collaborative filtering techniques, state-of-the art matrix factorization methods still has several drawbacks: (1) uniform or context-insensitive prediction; (2) over-fitting; (3) inefficiency; (4) ineffectiveness due to sparsity. This paper studies how to use hierarchical and local matrix decomposition to address these issues. The proposed algorithm Random Partition Matrix Factorization (RPMF) works by applying a set of local decomposition processes on sub-rating matrices. Thus, the users and items in the same local neighborhood receive higher impacts from each other than the rest of the items and users. To decompose the rating matrix, we explored a basic MF model to factorize the rating matrix, and then we have adopted a random partition approach to use decision tree to group similar users and items. To avoid over-fitting, RPMF generates an ensemble. In addition, we show that the proposed algorithm is a model-averaging gradient boosting model. It exploits boosting technique to reduce bias while using model-averaging to reduce the variance further. Formal analysis demonstrates that: (1) the error of the low-rank matrix approximation is bounded; (2) a single tree in RPMF is a generalized additive model as well as a boosting algorithm. As a result, it has good generalizability; (3) ensemble of RPMF further reduces the prediction error. Empirical studies have used several well-known benchmark datasets, including Movielens, Eachmoive, Netflix and Yahoo! music recommendation. The results demonstrate that the proposed method decreases RMSE of state-of-the-art matrix factorization methods, i.e., SVD++, BPMF and GP-LVM, by as much as 0.04, 0.02 and 0.03 respectively. Additional experiments show that RPMF decomposes the rating matrix locally to adjust different contents, and reduces the effects of over-fitting significantly.

We notice that RPMF is a general framework. First, different MF models can replace the basic MF at each node to obtain more accurate predictions, such as SVD++ [11]. Second, it is not only suitable for collaborative filtering but also other relational learning tasks, such as tag recommendation system. Third, it can utilize meta features, e.g, users' (items') attributes and temporal information. For example, one can use these features to regulate the factorization or to partition rating matrix instead of using latent factors at each node. Additionally, collective MF can also be applied at each node to utilize auxiliary knowledge from other domains. Thus, in the future, we plan to extend the proposed method to other related applications with rich sources of information and auxiliary domains, and compare against other context-aware approaches.

References

[1] Bin Cao, Nathan Nan Liu, and Qiang Yang. Transfer learning for collective link prediction in multiple heterogenous domains. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning*, ICML '10, pages 159–166, Haifa, Israel, 2010. Omni Press.

[2] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The yahoo! music dataset and kdd-cup'11. In *Proceedings of KDDCup 2011*, 2011.

[3] Wei Fan, Joe McCloskey, and Philip S. Yu. A general framework for accurate and fast regression by data summarization in random decision trees. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 136–146, New York, NY, USA, 2006. ACM.

[4] Wei Fan, Haixun Wang, Philip S. Yu, and Sheng Ma. Is random model better? on its accuracy and efficiency. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, ICDM '03, pages 51–58, Washington, DC, USA, 2003. IEEE Computer Society.

[5] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[6] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[7] Nicolas Gillis and François Glineur. A multilevel approach for nonnegative matrix factorization. *CORE Discussion Papers*, abs/1009.0881, 2010.

[8] Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[9] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 135–142, New York, NY, USA, 2010. ACM.

[10] Adam Kalai. Efficient estimators for generalized additive models, 2005.

[11] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[12] Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 601–608, New York, NY, USA, 2009. ACM.

[13] Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 617–624, New York, NY, USA, 2009. ACM.

[14] Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 203–210, New York, NY, USA, 2009. ACM.

[15] Lester W. Mackey, David Weiss, and Michael I. Jordan. Mixed membership matrix factorization. In *Proceedings of the 27th International Conference on Machine Learning*, ICML '10, pages 711–718, Haifa, Israel, 2010. Omni Press.

[16] Shie Mannor and Ron Meir. Geometric bounds for generalization in boosting. In *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, COLT '01, pages 461–472, London, UK, 2001. Springer-Verlag.

[17] Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 141–149, New York, NY, USA, 2011. ACM.

[18] A. Murua. Upper bounds for error rates of linear combinations of classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:591–602, 2002.

[19] Michael Peter Perrone. *Improving Regression Estimation: Averaging methods for variance reduction with extensions to general convex measure optimization*. PhD thesis, 1993.

[20] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 880–887, New York, NY, USA, 2008. ACM.

[21] Y. Seldin and N. Tishby. Pac-bayesian analysis of co-clustering and beyond. *Journal of Machine Learning Research*, 11:3595–3646, 2010.

[22] Ohad Shamir and Shai Shalev-Shwartz. Collaborative filtering with the trace norm: Learning, bounding, and transducing. In *Proceedings of the 24th Annual Conference on Learning Theory*, COLT '11, London, UK, 2011. Springer-Verlag.

[23] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 650–658, New York, NY, USA, 2008. ACM.

[24] Nathan Srebro, Noga Alon, and Tommi Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In *Proceedings of the 17th conference on Advances in Neural Information Processing Systems*, NIPS '08, pages 1321–1328, Cambridge, MA, 2004. MIT Press.

[25] Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *Proceedings of the 18th Annual Conference on Learning Theory*, COLT '05, pages 545–560, London, UK, 2005. Springer-Verlag.