Discrete Matrix Factorization and Extension for Fast Item Recommendation

Defu Lian, Xing Xie and Enhong Chen

Abstract—Binary representation of users and items can dramatically improve efficiency of recommendation and reduce size of recommendation models. However, learning optimal binary codes for them is challenging due to binary constraints, even if squared loss is optimized. In this article, we propose a general framework for discrete matrix factorization based on discrete optimization, which can 1) optimize multiple loss functions; 2) handle both explicit and implicit feedback datasets; 3) take auxiliary information into account without any hyperparameters. To tackle the challenging discrete optimization problem, we propose block coordinate descent based on semidefinite relaxation of binary quadratic programming. We theoretically show that it is equivalent to discrete coordinate descent when only one coordinate is in each block. We extensively evaluate the proposed algorithms on 8 real-world datasets. The results of evaluation show that they outperform the state-of-the-art baselines significantly and that auxiliary information of items improves recommendation performance. For better showing the advantages of binary representation, we further propose a two-stage recommender system, consisting of an item-recalling stage and a subsequent fine-ranking stage. Its extensive evaluation shows hashing can dramatically accelerate item recommendation with little degradation of accuracy.

Index Terms—Item Recommendation, Hashing, Block Coordinate Descent, Discrete Optimization, Two-stage Recommender Systems.

1 Introduction

The growing number of products and user evolving interest challenge instant item recommendation. Hashing is a very promising approach to address this challenge, which has been applied in Google News Recommendation [1]. The core idea of hash is to represent users and/or items by binary vectors (also called binary codes or hash codes) so that top-K preferred items can be retrieved very efficiently. Being used for recalling potentially preferred items, hashing can dramatically accelerate the recommendation of many complex algorithms [2], [3], [4], [5], [6], [7], since they only need to re-rank the recalled items.

In this article, we will study discrete matrix factorization (DMF) since matrix factorization methods are very efficient and effective for item recommendation [8]. Given an $m \times n$ user-item rating/preference matrix, DMF maps m users and n items into the same k-dimensional binary hamming space. Each user and item are then represented by k-bit binary codes. Compared to the real-valued model, the model size is then reduced by $31/32 \approx 96.9\%$. The dot product between binary codes estimates rating or preference, and can be computed very efficiently via hamming distance (using CPU instructions __popcnt and xor) [9]. Given a user's binary code, it is possible to exactly retrieve the top-K preferred items from n candidate items in sublinear time via multi-index hashing [10]. Approximated retrieval can be even done in logarithmic or even constant time [11], [12].

Learning optimal binary codes is generally NP-hard [13] due to binary constraints. To address this challenge, several heuristic algorithms [14], [15], [16] have been proposed, which first solves a relaxed optimization algorithm via discarding the binary constraints, and then quantizes the relaxed solutions. However, Zhang et al. observed that these heuristic algorithms resulted in a large quantization loss [17] and proposed to learn binary codes directly. Nevertheless, several important issues are not addressed yet. First, their proposed algorithm is based on rating-based prediction, instead of item-based recommendation. It is well known that item-based recommendation is the ultimate goal of practical recommender systems [18] and that optimizing rating-based metrics cannot guarantee better item-based recommendation performance. Second, their proposed algorithm is optimized via cyclic coordinate descent, which only updates one bit each time. This is suboptimal for a combinatorial optimization problem. Third, only rating data is modeled, but there are other types of feedback (implicit feedback such as click, view and like) in recommender systems. Fourth, auxiliary information of users and items cannot be taken into account, thus cold-start problems cannot be well handled. It is worth noting that multi-modal hashing is very challenging and simple aggregation may not work well. Finally, though the efficiency improvement induced by binary representation has been reported in [16], it is unclear how to balance the efficiency and effectiveness of recommender systems, since hashing inevitably incurs quantization loss.

To this end, we first propose a general framework of discrete matrix factorization. It can take both explicit and implicit feedback as input, by supporting squared loss and logistic loss. Whatever loss function is used, the optimization of this framework can be boiled down to a binary quadratic programming (BQP) problem. To avoid bad local optimum, it is better to solve BQP via semidefinite relaxation (SDR) followed by Gaussian randomization (GR). SDR

- *Defu Lian and Enhong Chen are with Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui. Email:{liandefu,cheneh}@ustc.edu.cn.*
- *Xing Xie is with Microsoft Research Asia, Beijing, China. Email: xingx@microsoft.com.*

is known as an excellent solver of BQP [19], but suffers from computational issues. In particular, regarding k-variable BQP with a positive semidefinite matrix, SDR+GR induces less than 36.4% degradation of the objective value, but costs $\mathcal{O}(k^{3.5})$ for fixed precision [19]. This motivates the proposal of block coordinate descent (BCD), compromising between cyclic coordinate descent and SDR. Moreover, it will be theoretically shown that BCD subsumes cyclic coordinate descent when only one coordinate is in each block. To better support item-based recommendation, an interaction-based regularization is introduced, to penalize users' non-zero estimated preference for unrated items. This also helps to deal with the sparsity challenge of recommender systems, particularly in implicit feedback. For the sake of deriving more compact and informative binary codes, balanced and decorrelated constraints are also imposed, as suggested in [17]. To incorporate auxiliary information of users and items, we extend this framework by introducing extra hash codes for auxiliary information and propose to learn hash codes for users and items with precluding encoded knowledge in extra codes. Interestingly, the fusion is hyperparameter-free, providing some insights for multi-modal hashing. To better show the advantages of binary representation, we further propose a two-stage recommender system, consisting of a highly efficient item-recalling stage and a highly accurate fine-ranking stage. The item-recalling stage utilizes discrete matrix factorization, since it can dramatically accelerate the retrieval of the top-K preferred items.

Finally, we extensively evaluated the proposed framework on 8 real-world datasets. The results show that the proposed algorithms outperform the state-of-the-art baselines significantly. The proposed BCD-based optimization can yield much lower objective values and significant higher recommendation performance than cyclic coordinate descent. In spite of hyperpameter-free, the extension for auxiliary information performs much better than competing baselines, revealing that simply aggregation can not deal with multi-modal hashing. The evaluation of the two-stage recommender system shows that discrete matrix factorization can significantly accelerate item recommendation with less than 4% degradation of NDCG@20 in most cases.

This paper is an extension of our preliminary paper [20], in which we incorporate auxiliary information of users and items by a regression-based modeling, consider the logistic loss function and introduce an interaction regularization. In this article, we further deliver the following contributions.

- We propose a block coordinate descent (BCD) to directly learn binary codes for compromising the efficiency and effectiveness of optimization. It is theoretically shown that BCD subsumes cyclic coordinate descent when each coordinate is considered a block. The evaluation results show that the optimization algorithm yields lower loss value and higher recommendation performance.
- We propose a hyperparameter-free method to model auxiliary information. The evaluation on multiple datasets shows that it is much better at retrieving potentially preferred items than the regression-based method and other baselines.
- We propose a two-stage recommender system, consisting of highly efficient item-recalling stage and highly accurate fine-ranking stage. The evaluation results better show the advantage of discrete matrix factorization in accelerating the recommendation of practical recommender systems.
- We conduct more extensive experiments on more and larger-scale datasets, report more performance metrics of recommendation, and compare the proposed algorithms with more competing baselines. The evaluation results better show the significant superiority of the proposed algorithms.

2 Related Work

We will review recent advance of hashing-based recommendation algorithms. For comprehensive reviews of hashing techniques, please refer to [21], [22]. For better presentation, we organize them into two categories: quantization-based methods, which first solve relaxed problems and then quantize relaxed solutions, and optimization-based methods, which directly tackle optimization with binary constraints.

2.1 Quantization-based Methods

Hashing can be categorized into data-independent or data-dependent methods. The pioneer work of hashing in recommender systems is to use data-independent hashing, i.e., Locality Sensitive Hashing (LSH), for clustering similar Google News readers, so as to find similar users efficiently [1]. Then a two-stage framework was proposed for scalable news recommendation [23], by building hierarchical clustering on the results of LSH. However, it is totally different from our two-stage framework, which learns hash codes from rating data and auxiliary information jointly. Data-dependent hashing, depending on the actual points in the dataset, attracts more attention in other scenarios of recommendation in recent years. For example, random projection was applied for mapping user/item latent factors from regularized matrix factorization into a hamming space [24]. Similarly, Zhou and Zha [15] exploited iterative quantization (ITQ) for generating binary codes from real-valued user/item latent factors. In order to derive compact hash codes, the decorrelated constraint was imposed on user/item real-valued latent factors [14] before quantization. However, because of the loss of latent factors' magnitudes induced by quantization, hashing only preserves similarity between user and item, rather than inner product based preference [16]. Therefore, Zhang et al. proposed to impose a constant feature-norm constraint on user/item real-valued latent factors, and then quantized magnitudes and similarity separately. The relevant work can be summarized as two independent steps: relaxed learning of real-valued latent factors with specific constraints, and subsequent binary discretization.

2.2 Optimization-based Methods

The two-step methods suffer from a large quantization loss according to [17]. Therefore, Zhang et al. proposed to directly learn binary codes in matrix factorization with binary constraints by cyclic coordinate descent [17]. For the sake of obtaining informative and compact hash codes, the balanced and decorrelated constraints were further imposed. However, this algorithm was only designed for rating data, but

not applicable for binary or implicit feedback. Hence, Zhang et al. proposed to optimize the pairwise ranking between interacted and non-interacted items for each user with binary constraints [25]. The optimization was also based on cyclic coordinate descent. Lian et al. proposed a unified framework for both explicit and implicit feedback datasets by the introduction of interaction regularization [20] and further incorporated auxiliary information by a regression-based method. Zhang et al. built Deep Belief Network for modeling auxiliary information and incorporated its representation into collaborative filtering [26]. Liu et al. proposed discrete factorization machine for better modeling auxiliary information [27] and learned hash codes also based on cyclic coordinate descent, but targeted rating data and only ranked rated items.

Compared to existing research, the proposed algorithm has the following differences (also advantages) in addition to the ones[1] inherited from the preliminary work [20]. First, the optimization problem is based on block coordinate descent, which can subsume the widely-used cyclic coordinate descent. Second, the proposed method for modeling auxiliary information is hyperparameter-free, efficient and effective for item recommendation. Finally, we propose a two-stage recommender system, better showing the advantages of binary representation in practical recommendation.

3 Preliminary

Explicit feedback is usually represented by a rating matrix while implicit feedback by a preference matrix. They are collectively referred as a rating matrix in the sequel. Assume there are m users and n items, then the rating matrix $\boldsymbol{R}$ is of size $m \times n$, and its each non-zero entry r_{ij} indicates a user's rating/preference for an item j. The observed entries are denoted by $\Omega = \{(i,j)|r_{ij} \text{ is known}\}$. The set of items which a user i rates is denoted by $\mathbb{I}_i$ and the set of users rating an item j by $\mathbb{U}_j$.

3.1 Matrix Factorization with Priors on Missing Values

Weighted Regularized Matrix Factorization (WRMF) is a very effective method for collaborative filtering for implicit feedback [28] and optimizes the following objective function

$$\sum_{i,j} w_{ij}(r_{ij} - \boldsymbol{p}_i^T \boldsymbol{q}_j)^2 + \lambda(\sum_i \|\boldsymbol{p}_i\|^2 + \|\sum_j \boldsymbol{q}_j\|^2),$$

where w_{ij} denotes the weight of the rating r_{ij}, $\boldsymbol{p}_i$ and $\boldsymbol{q}_j \in \mathbb{R}^k$ represent latent factors of user i and item j, respectively. w_{ij} is set to 1 if $(i,j) \notin \Omega$ and to a tunable parameter α ($\gg 0$) otherwise according to [28]. This weight is also considered the confidence of the rating, so ratings have much higher confidence than others. This first part of the objective function can be decomposed into two components,

$$\sum_{i,j} w_{ij}(r_{ij} - \boldsymbol{p}_i \boldsymbol{q}_j)^2 = \alpha \sum_{(i,j)\in\Omega} (r_{ij} - \boldsymbol{p}_i^T \boldsymbol{q}_j)^2 + \sum_{(i,j)\notin\Omega} (\boldsymbol{p}_i^T \boldsymbol{q}_j)^2,$$

where $r_{ij} = 0$ if $(i,j) \notin \Omega$. Therefore, in addition to classical ℓ_2-norm, WRMF intrinsically imposes an interaction regularization on missing values, which penalizes non-zero preference prediction of unrated items for each user. It is very nature to extend the regularization to rating data, as done by [29], and the effect for improving item recommendation was empirically studied on multiple datasets. The interaction regularization, also dubbed implicit regularization [30], has also been extended to tensor factorization. We will also adopt this interaction regularization for handling both explicit feedback and implicit feedback.

1. Handling multiple loss function, taking both explicit and implicit feedback as input

3.2 Binary Quadratic Programming

Binary quadratic programming (BQP) is a classical combinatorial optimization problem [19], which minimizes a quadratic function with respect to binary variables, i.e.,

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}, \text{ s.t. } x_d^2 = 1, d = 1 \cdots, k \tag{1}$$

where $\boldsymbol{A}$ is a real square matrix. The BQP is well-known as a computationally difficult problem, particularly belonging to the class of NP-hard problems. Computing good solutions is quite a difficult task.

Semidefinite relaxation (SDR) technique is a powerful computationally efficient approximation technique for many difficult optimization problems including binary quadratic programming. A crucial step in deriving a SDR of BQP is to observe that $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = \text{trace}(\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^T)$. Introducing a new variable $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$ and noting that $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$ is equivalent to $\boldsymbol{X}$ being a rank-one positive semidefinite (PSD) matrix, we can obtain the following equivalent formulation

$$\begin{aligned} &\min_{\boldsymbol{X}\in\mathbb{S}_+^k} \text{trace}(\boldsymbol{A}\boldsymbol{X}) \\ &\text{s.t. } X_{d,d} = 1, \text{rank}(\boldsymbol{X}) = 1, \end{aligned}$$

where $\mathbb{S}_+^k$ denote the set of PSD matrices of size $k \times k$. The reformulation allows us to identify the fundamental difficulty of solving BQP, i.e., $\text{rank}(\boldsymbol{X}) = 1$. Thus, we may drop it to obtain the following semidefinite relaxation of BQP:

$$\min_{\boldsymbol{X}\in\mathbb{S}_+^k} \text{trace}(\boldsymbol{A}\boldsymbol{X}), \text{ s.t. } \text{diag}(\boldsymbol{X}) = \boldsymbol{1}_k. \tag{2}$$

SDR can then be solved, to any arbitrary accuracy (ϵ), in a numerically reliable and efficient fashion, using customize interior-point algorithms [31] with time complexity of $\mathcal{O}(k^{3.5}\log(1/\epsilon))$. However, it remains unknown how to convert the optimum of Eq (2) into a feasible solution of Eq (1).

If the optimal $\boldsymbol{X}^*$ is of rank one, then $\boldsymbol{X}^* = \boldsymbol{x}^*\boldsymbol{x}^{*T}$, and $\boldsymbol{x}^*$ is the optimal solution of BQP. Otherwise, Gaussian randomization procedure can be applied, as shown in Algorithm 1, since $\boldsymbol{X}^*$ is also the optimal solution to the following stochastic BQP problem,

$$\min_{\boldsymbol{X}\in\mathbb{S}_+^n} E_{\xi\sim N(\boldsymbol{0},\boldsymbol{X})}[\boldsymbol{\xi}^{\boldsymbol{T}} \boldsymbol{A} \boldsymbol{\xi}], \text{ s.t. } E_{\boldsymbol{\xi}\sim N(\boldsymbol{0},\boldsymbol{X})}[\xi_i^2] = 1.$$

According to [32], the approximation accuracy of the SDR followed by Gaussian randomization is no worse than $\frac{2}{\pi} \approx 0.63661$ when $\boldsymbol{A}$ is PSD. In other words, even though BQP is NP-hard, this algorithm can obtain a solution whose objective value is at most $\frac{2}{\pi}$ times the optimal value of SDR.

Algorithm 1: Round($\boldsymbol{X}$): Gaussian Randomization

Input: The SDR solution $\boldsymbol{X}^* \in \mathbb{R}^{k\times k}$
Output: $\boldsymbol{x}^*$ the best approximated feasible solution
for $\ell = 1, 2, \cdots, k$ **do**
  generate $\boldsymbol{\xi}_\ell \sim N(0, \boldsymbol{X}^*)$;
  construct a feasible point $\boldsymbol{x}_\ell \leftarrow \text{sign}(\boldsymbol{\xi}_\ell)$;
determine $\ell^* \leftarrow \arg\min_{\ell\in\{1,\cdots,k\}} \boldsymbol{x}_\ell^T \boldsymbol{A}\boldsymbol{x}_\ell$;
$\boldsymbol{x}^* \leftarrow \boldsymbol{x}_{\ell^*}$ as the best approximated solution of BQP;

4 Discrete Matrix Factorization and Extension

The proposed algorithm treats binary codes as parameters and learns codes directly by optimization algorithms. Denoting $\boldsymbol{\phi}_i \in \{\pm 1\}^k$ binary code of user i and $\boldsymbol{\psi}_j \in \{\pm 1\}^k$ binary code of item j, the inner product is $\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j = k - 2\text{H}(\boldsymbol{\phi}_i, \boldsymbol{\psi}_j)$, where $\text{H}(\boldsymbol{\phi}, \boldsymbol{\psi})$ denotes the hamming distance between binary codes. Based on CPU instructions (xor and __popcnt), hamming distance between binary codes is extremely efficient. Below, we elaborate how to directly learn binary codes for users and items.

4.1 Loss Function

Let's first figure out the loss function. As mentioned above, we need to introduce the interaction regularization. Moreover, we should take both explicit and implicit feedback as inputs. Therefore, the loss function for discrete matrix factorization is formulated as follows,

$$\begin{aligned}\mathcal{L} = &\sum_{(i,j)\in\Omega} \ell(r_{ij}, \boldsymbol{\phi}_i^T\boldsymbol{\psi}_j) + \rho \sum_{(i,j)\notin\Omega} (\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j)^2 \\ &\text{s.t. } \boldsymbol{\phi}_i \in \{\pm 1\}^k, \boldsymbol{\psi}_j \in \{\pm 1\}^k,\end{aligned} \tag{3}$$

where $\ell(r_{ij}, \hat{r}_{ij})$ is a convex loss function, which can be square loss $\ell(r_{ij}, \hat{r}_{ij}) = (r_{ij} - \hat{r}_{ij})^2$, and logistic loss $\ell(r_{ij}, \hat{r}_{ij}) = \log(1 + e^{-r_{ij}\hat{r}_{ij}})$ in this article. Note that logistic loss can be used for binary feedback or implicit feedback [33]. Therefore, the capability of modeling both explicit and implicit feedback datasets benefits from the usage of different loss functions and introduction of interaction regularization.

To ensure binary codes compact and informative, the balanced and decorrelated constraints are usually imposed so that shorter code can encode more information [34]. If $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_m]^T$ and $\boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_n]^T$, the balanced constraints are $\mathbf{1}^T\boldsymbol{\Phi} = 0, \mathbf{1}^T\boldsymbol{\Psi} = 0$, and the decorrelated constraints are $\boldsymbol{\Phi}^T\boldsymbol{\Phi} = m\boldsymbol{I}_k, \boldsymbol{\Psi}^T\boldsymbol{\Psi} = n\boldsymbol{I}_k$. Since Eq (3) itself is a combinatorial optimization problem, these constraints make this optimization much more challenging. Previous work usually softened these constraints by introducing real-valued delegate variables with those constraints and minimizing the distance between binary codes and real-valued delegations. However, the effect of balance may be different from decorrelation, so the joint modeling may not give play to their respective advantages. Therefore, we suggest the following two regularizations to impose these two constraints separately,

$$\begin{aligned}\mathcal{R}_1 = &\|\boldsymbol{\Phi} - \boldsymbol{B}_b\|_F^2 + \|\boldsymbol{\Psi} - \boldsymbol{D}_b\|_F^2, \\ &\text{s.t. } \mathbf{1}^T\boldsymbol{B}_b = \mathbf{0} \text{ and } \mathbf{1}^T\boldsymbol{D}_b = \mathbf{0}\end{aligned} \tag{4}$$

and

$$\begin{aligned}\mathcal{R}_2 = &\|\boldsymbol{\Phi} - \boldsymbol{B}_d\|_F^2 + \|\boldsymbol{\Psi} - \boldsymbol{D}_d\|_F^2, \\ &\text{s.t. } \boldsymbol{B}_d^T\boldsymbol{B}_d = m\boldsymbol{I}_k \text{ and } \boldsymbol{D}_d^T\boldsymbol{D}_d = n\boldsymbol{I}_k\end{aligned} \tag{5}$$

where $\boldsymbol{B}_b$ and $\boldsymbol{B}_d$ are delegations of $\boldsymbol{\Phi}$ for balanced and decorrelated constraints respectively, and $\boldsymbol{D}_b$ and $\boldsymbol{D}_d$ are delegations of $\boldsymbol{\Psi}$. Therefore, discrete matrix factorization optimizes the loss function:

$$\min_{\Phi,\boldsymbol{B}_b,\boldsymbol{B}_d,\Psi,\boldsymbol{D}_b,\boldsymbol{D}_d} \mathcal{L} + \alpha\mathcal{R}_1 + \beta\mathcal{R}_2. \tag{6}$$

4.2 Optimization

We follow alternating optimization for optimizing the loss in Eq (6). Alternating optimization takes turn learning parameters iteratively. Being independent to $\ell(\cdot,\cdot)$, the objective functions with respect to $\boldsymbol{B}_b, \boldsymbol{B}_d, \boldsymbol{D}_b, \boldsymbol{D}_d$ are easy to optimize but elaborated at the end of this section. In case of squared loss, the objective functions related to $\boldsymbol{\phi}_i$ and $\boldsymbol{\psi}_j$ correspond to binary quadratic programming problems, but inhomogeneous[2]. Due to the symmetry of $\boldsymbol{\phi}_i$ and $\boldsymbol{\psi}_i$, below we only take $\boldsymbol{\phi}_i$ as an example. The objective function with respect to $\boldsymbol{\phi}_i$ is formulated as,

$$\begin{aligned}\min_{\boldsymbol{\phi}_i\in\{\pm 1\}^k} &\boldsymbol{\phi}_i^T\big((1-\rho)\boldsymbol{\Psi}_i^T\boldsymbol{\Psi}_i + \rho\boldsymbol{\Psi}^T\boldsymbol{\Psi}\big)\boldsymbol{\phi}_i \\ &- 2\boldsymbol{\phi}_i^T\big(\boldsymbol{\Psi}_i^T\boldsymbol{r}_i + \alpha\boldsymbol{B}_b(i,:) + \beta\boldsymbol{B}_d(i,:)\big)\end{aligned} \tag{7}$$

where $\boldsymbol{\Psi}_i$ is a submatrix of $\boldsymbol{\Psi}$ which only includes rows of rated items by the user i, and $\boldsymbol{r}_i$ is his/her rating vector over rated items. $\boldsymbol{B}(i,:)$ is denoted the i-th row of matrix $\boldsymbol{B}$.

In case of logistic loss, in spite of non-linearity, the objective function can also be boiled down to inhomogeneous binary quadratic programming problems by seeking a upper variational quadratic bound [35]. In particular, the upper bound of $\log(1 + e^{-r_{ij}\boldsymbol{\phi}_i'\boldsymbol{\psi}_j})$, $r_{ij} \in \{\pm 1\}$, can be obtained based on the following inequality,

$$\begin{aligned}&\log(1 + e^{-r_{ij}\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j}) \\ =&\log(1 + e^{\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j}) - \frac{1 + r_{ij}}{2}\boldsymbol{\phi}_i'\boldsymbol{\psi}_j \\ \le&\lambda(\hat{r}_{ij})\big((\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j)^2 - \hat{r}_{ij}^2\big) - \frac{1}{2}(r_{ij}\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j + \hat{r}_{ij}) + \log(1 + e^{\hat{r}_{ij}})\end{aligned}$$

where $\lambda(x) = \frac{1}{4x}\tanh(x/2) = \frac{1}{2x}(\sigma(x) - \frac{1}{2})$ and the equality holds only if $\hat{r}_{ij} = \boldsymbol{\phi}_i^T\boldsymbol{\psi}_j$. Note that $\lambda(0) = \frac{1}{8}, \lambda(x) = \lambda(-x)$. Based on this upper bound, the objective function for user i is to solve

$$\begin{aligned}\min_{\boldsymbol{\phi}_i\in\{\pm 1\}^k} &\boldsymbol{\phi}_i^T\big(\boldsymbol{\Psi}_i^T\text{diag}(\boldsymbol{\lambda}_i - \rho)\boldsymbol{\Psi}_i + \rho\boldsymbol{\Psi}^T\boldsymbol{\Psi}\big)\boldsymbol{\phi}_i \\ &- 2\boldsymbol{\phi}_i^T\big(\boldsymbol{\Psi}_i^T\boldsymbol{r}_i/4 + \alpha\boldsymbol{B}_b(i,:) + \beta\boldsymbol{B}_d(i,:)\big)\end{aligned} \tag{8}$$

where $\boldsymbol{\lambda}_i = [\lambda(\hat{r}_{ij})]_{j\in\mathbb{I}_i}$. In fact, the quadratic upper bound exists for any convex loss functions with Lipschitz continuous gradient, so this technique can be applied to many other convex loss functions. It is worth noting that we need to keep $\{\hat{r}_{ij}|(i,j)\in\Omega\}$ up-to-date after updating $\boldsymbol{\phi}_i$ in this case. Below we study how to solve the following inhomogeneous binary quadratic programming problem,

$$\min_{\boldsymbol{x}\in\{\pm 1\}^k} \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{b}^T\boldsymbol{x}, \tag{9}$$

where $\boldsymbol{A}$ is positive semidefinite.

2. Note that rating should be scaled to align with the range of preference estimation [17].

4.2.1 Cyclic Coordinate Descent

As we investigated, most existing work of hashing-based recommendation uses cyclic coordinate descent (CCD) for optimizing binary quadratic programming problems. CCD learns one bit each time given other bits fixed, so that the bit is optimally updated each time. Assuming the bit x_d to be optimized, the objective function is then reformulated as

$$\min_{x_d\in\{\pm 1\}} a_{dd}x_d^2 + 2x_d(\boldsymbol{a}_d^T\boldsymbol{x} - a_{dd}x_d) - 2b_dx_d, \tag{10}$$

where $a_{dd} = A(d,d)$ and $\boldsymbol{a}_d$ the d-th row of the matrix $\boldsymbol{A}$. This objective function is equivalent to

$$\min_{x_d\in\{\pm 1\}} x_d(\boldsymbol{a}_d^T\boldsymbol{x} - b_d). \tag{11}$$

Then the optimal x_d is obtained by

$$x_d^\star = \begin{cases} 1, & \text{if } \boldsymbol{a}_d^T\boldsymbol{x} - b_d < 0 \\ -1, & \text{if } \boldsymbol{a}_d^T\boldsymbol{x} - b_d > 0 \\ \text{unchanged}, & \text{otherwise} \end{cases} \tag{12}$$

Note that x_d keeps unchanged when $\boldsymbol{a}_d^T\boldsymbol{x} = b_d$, implying that it is very important to set a good initialization to $\boldsymbol{x}$. The updating rule is applied for each bit iteratively until convergence (e.g. $\boldsymbol{x}$ doesn't change any more).

4.2.2 Semidefinite Relaxation

Section 3.2 only discusses semidefinite relaxation of homogeneous binary quadratic programming. Here we discuss inhomogeneous ones. We first introduce a variable $t \in \{\pm 1\}$, and consider the problem,

$$\min_{\boldsymbol{x},t} [\boldsymbol{x}^T, t] \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{b} \\ -\boldsymbol{b}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix}, \tag{13}$$

which can be solved by the aforementioned techniques, i.e., SDR followed by Gaussian randomization. Assuming $[\tilde{\boldsymbol{x}}^\star, t^\star]$ is an optimal solution to this problem, $\boldsymbol{x}^\star = \tilde{\boldsymbol{x}}^\star \times t^\star$ is the optimal solution of the problem in Eq (9). This equivalence can be observed by rewriting this problem as $\min_{\boldsymbol{x},t} \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - 2t\boldsymbol{b}^T\boldsymbol{x}$. More particularly, if $t^\star = 1$, $\tilde{\boldsymbol{x}}^\star$ is the optimum of Eq (9); if $t^\star = -1$, $-\tilde{\boldsymbol{x}}^\star$ is the optimum of Eq (9).

4.2.3 Block Coordinate Descent

Although SDR is known as an excellent solver for binary quadratic programming, it suffers from computational issues and its time complexity is up to $\mathcal{O}(k^{3.5})$ for fixed precision. Therefore, we strike a balance between accuracy (SDR) and efficiency (CCD), by proposing a block coordinate descent (BCD) algorithm for optimization. Assuming the block coordinate to optimize is denoted by $\boldsymbol{x}_l \in \{\pm 1\}^{k_1}$ of a block l, and splitting the $\boldsymbol{A}$, $\boldsymbol{x}$ and $\boldsymbol{b}$ according to the dependence of $\boldsymbol{x}_l$, the problem in Eq (9) can be reformulated as

$$\min_{\boldsymbol{x}_l\in\{\pm 1\}^{k_1}} [\boldsymbol{x}_l^T, \boldsymbol{x}_{\bar{l}}^T] \begin{bmatrix} \boldsymbol{A}_{(l,l)} & \boldsymbol{A}_{(l,\bar{l})} \\ \boldsymbol{A}_{(\bar{l},l)} & \boldsymbol{A}_{(\bar{l},\bar{l})} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_l \\ \boldsymbol{x}_{\bar{l}} \end{bmatrix} - 2[\boldsymbol{b}_l^T, \boldsymbol{b}_{\bar{l}}^T] \begin{bmatrix} \boldsymbol{x}_l \\ \boldsymbol{x}_{\bar{l}} \end{bmatrix}.$$

Due to the symmetry of $\boldsymbol{A}$, $\boldsymbol{A}_{(l,\bar{l})}^T = \boldsymbol{A}_{(\bar{l},l)}$. Ignoring the terms independent to $\boldsymbol{x}_l$, the objective function is then equivalent to

$$\min_{\boldsymbol{x}_l\in\{\pm 1\}^{k_1}} \boldsymbol{x}_l^T\boldsymbol{A}_{(l,l)}\boldsymbol{x}_l - 2\boldsymbol{x}_l^T(\boldsymbol{b}_l - \boldsymbol{A}_{(l,\bar{l})}\boldsymbol{x}_{\bar{l}}). \tag{14}$$

The problem size can be much smaller than the original problem in Eq (9) so that efficiency can be dramatically improved. Compared to CCD, this problem learns $\boldsymbol{x}_l$ jointly so that the approximation accuracy can be improved. Note that $\boldsymbol{A}_{(l,l)}$, i.e., the principle submatrix of PSD $\boldsymbol{A}$, is also PSD, this returns to the inhomogeneous BQP.

Due to Gaussian randomization, it is still unknown how BCD correlates with CCD when only one coordinate is in each block. The following theorem answers this question.

Lemma 1. *When $k_1 = 1$, $\boldsymbol{x}$ and $\boldsymbol{b}$ and $\boldsymbol{A}$ are scalars in Eq (13), denoted by x, b, a respectively, the optimum of its SDR is*

$$(xt)^\star = \begin{cases} 1 & b > 0 \\ -1 & b < 0 \\ 0 & b = 0 \end{cases} \tag{15}$$

Proof. When $k_1 = 1$, Eq (2) can be rewritten to

$$\min_{x,t} a - 2bxt, \text{ s.t. } -1 \le xt \le 1$$

Then the optimum of xt depends on the sign of b and is yielded in Eq (15). Note that when $b = 0$, the objective function is constant, so the optimum is the initial value. Here $\boldsymbol{X}$ in Eq (2) is assumed to be initialized as an identity matrix. □

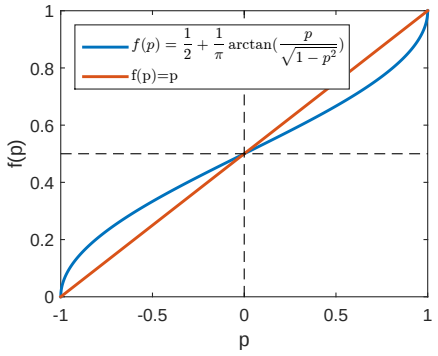
Lemma 2. *If* $\begin{bmatrix} x \\ y \end{bmatrix} \sim N(\boldsymbol{0}, \begin{bmatrix} 1 & p \\ p & 1 \end{bmatrix})$, *then*

$$P(xy > 0) = \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{p}{\sqrt{1-p^2}}).$$

Proof. $P(xy > 0)$

$$\begin{aligned}
&= \int_0^\infty \int_0^\infty \frac{1}{\pi\sqrt{1-p^2}} \exp\big(-\frac{x^2+y^2-2pxy}{2(1-p^2)}\big)dxdy \\
&= \frac{1}{\pi}\int_0^\infty \int_0^\infty \sqrt{1-p^2}\exp\big(-\frac{1}{2}(x^2+y^2-2pxy)\big)dxdy \\
&= \frac{1}{\pi}\int_0^\infty \int_{-\frac{p}{\sqrt{1-p^2}}v}^\infty \exp\big(-\frac{1}{2}(u^2+v^2)\big)dudv \\
&= \frac{1}{\pi}\int_0^\infty \int_{-\arctan(\frac{p}{\sqrt{1-p^2}})}^{\frac{\pi}{2}} \exp(-\frac{r^2}{2})rdrd\theta \\
&= \frac{1}{\pi}(\frac{\pi}{2} + \arctan(\frac{p}{\sqrt{1-p^2}})) \\
&= \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{p}{\sqrt{1-p^2}})
\end{aligned}$$

□



Theorem 1. *CCD is almost equivalent to BCD when only one coordinate is in each block.*

Proof. It is easy to show that Gaussian randomization follows $\begin{bmatrix}\tilde{x}\\ t\end{bmatrix} \sim N(\mathbf{0}, \begin{bmatrix}1 & (xt)^\star\\ (xt)^\star & 1\end{bmatrix})$ when $k_1 = 1$. Lemma 2 implies $P(x=1) = P(\tilde{x}t > 0) = \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{(xt)^\star}{\sqrt{1-((xt)^\star)^2}})$ According to Lemma 1, $x = \text{sign}(b)$ if $b \neq 0$ and $P(x=1) = 0.5$ otherwise. Therefore, solving Eq (10) via SDR yields the same solution if $b_d - \boldsymbol{a}_d^T\boldsymbol{x} \neq 0$. The differences lies in the rare case of $b_d - \boldsymbol{a}_d^T\boldsymbol{x} = 0$, where BCD performs uniform sampling from $\{\pm 1\}$ while CCD remains it unchanged. □

Algorithm 2: BCD($\boldsymbol{A}, \boldsymbol{b}$): Block Coordinate Descent for BQP

Input: a psd matrix $\boldsymbol{A}$ and $\boldsymbol{b}$
Output: $\boldsymbol{x}^\star$ for $\min_{\boldsymbol{x}} \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{b}^T\boldsymbol{x}$

```
Initialize x;
repeat
    Randomly divide {1, ··· , k} into #bk blocks ;
    for l = {1, ··· , #bk} do
        X* ← Solve SDR of Eq (14) ;        // O(k_1^3.5)
        x_l ← Round(X*);                   // O(k_1^3)
until Convergent;
x* ← x;
```

The overall procedure of block coordinate descent for BQP is shown Algorithm 2 where k is assumed a multiple of $\#bk$. Note that random grouping is done in each iteration for the sake of better optimization. The time complexity is $\mathcal{O}(k_1^{2.5}k)$, where $k_1 = \frac{k}{\#bk}$. The empirical comparison between BCD and CCD for discrete matrix factorization on multiple datasets is illustrated in Fig. 1 and Fig. 2. They clearly show that BCD is much better than CCD at minimizing the loss function, though BCD costs more time. With the growing block size, the loss can be further reduced, but the running time also increases. As a result, block coordinate descent indeed strikes a balance between efficiency and effectiveness of optimization.

We next investigate how to learn delegate variables, including $\boldsymbol{B}_b$, $\boldsymbol{B}_d$, $\boldsymbol{D}_b$ and $\boldsymbol{D}_b$. The updating equation of $\boldsymbol{B}_b$ is similar to $\boldsymbol{D}_b$, and the updating equation of $\boldsymbol{B}_d$ is also similar to $\boldsymbol{D}_d$, so we only consider $\boldsymbol{B}_b$ and $\boldsymbol{B}_d$.

Updating $\boldsymbol{B}_b$. We use Lagrangian multiplier method to solve the following optimization

$$\min_{\boldsymbol{B}_b, \boldsymbol{\eta}} \|\boldsymbol{\Phi} - \boldsymbol{B}_b\|_F^2 + \mathbf{1}^T\boldsymbol{B}_b\boldsymbol{\eta}. \tag{16}$$

The optimal $\boldsymbol{\eta}^\star = \frac{2}{m}\boldsymbol{\Phi}^T\mathbf{1}$ and optimal $\boldsymbol{B}_b^\star = \boldsymbol{\Phi} - \frac{1}{2}\mathbf{1}(\boldsymbol{\eta}^\star)^T = (\boldsymbol{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T)\boldsymbol{\Phi}$. Therefore, the optimal $\boldsymbol{B}_b$ is the centralized $\boldsymbol{\Phi}$ since $\boldsymbol{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T$ is known as a centering operator. Substituting them back to Eq (16), this equals to

$$\|\frac{1}{n}\mathbf{1}\mathbf{1}^T\boldsymbol{\Phi}\|_F^2 = \boldsymbol{\Phi}^T\mathbf{1}\mathbf{1}^T\boldsymbol{\Phi} = \|\mathbf{1}^T\boldsymbol{\Phi}\|_F^2. \tag{17}$$

Therefore, the proposed regularization for balanced constraints is equivalent to $\|\mathbf{1}^T\boldsymbol{\Phi}\|^2$, constraining column-sum to be small.

Updating $\boldsymbol{B}_d$. The objective function for learning $\boldsymbol{B}_d$ is equivalent to

$$\max_{\boldsymbol{B}_d} \text{trace}(\boldsymbol{\Phi}^T\boldsymbol{B}_d), \text{ s.t. } \boldsymbol{B}_d^T\boldsymbol{B}_d = m\boldsymbol{I}_k \tag{18}$$

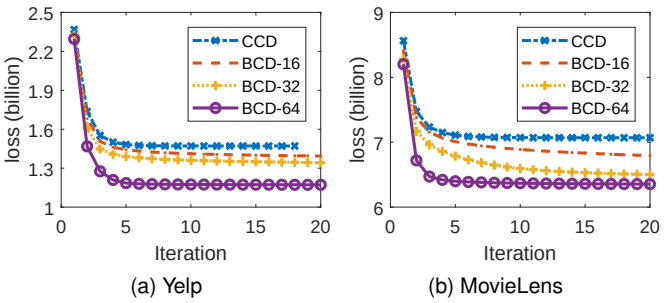


Fig. 1. Convergence curve of the overall objective function.

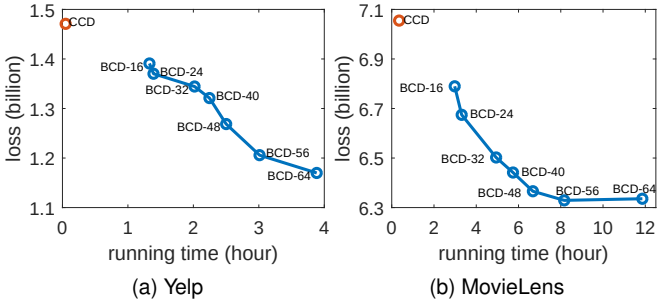


Fig. 2. Trade-off between Efficiency and Effectiveness.

Let $\tilde{\boldsymbol{B}}_d = \frac{1}{\sqrt{m}}\boldsymbol{B}_d$, then the problem is the same as the projection of a matrix onto the Stiefel manifold $\{\boldsymbol{A} \in \mathbb{R}^{m\times k} | \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_k\}$. The projection can be solved analytically [36]. In particular, let $\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the thin SVD of $\boldsymbol{\Phi}$, where each column of $\boldsymbol{U} \in \mathbb{R}^{m\times k}$ and $\boldsymbol{V} \in \mathbb{R}^{k\times k}$ corresponds to a left and right singular vector respectively, then the optimal $\boldsymbol{B}_d$ is given as follows:

$$\boldsymbol{B}_d^\star = \sqrt{m}\boldsymbol{U}\boldsymbol{V}^T. \tag{19}$$

The optimality can be easily established by the Von Neumann's trace inequality [37], that is $\text{trace}(\boldsymbol{A}\boldsymbol{B}) \leq \sum_i \sigma_i(\boldsymbol{A})\sigma_i(\boldsymbol{B})$, where $\sigma_i(\boldsymbol{A})$ is the i-th largest singular value of $\boldsymbol{A}$.

4.3 Initialization

Optimizing discrete matrix factorization is NP-hard, and an approximation algorithm is suggested. A good initialization algorithm is important for faster convergence and for approaching better local optimum. Here we first solve a relaxed problem of Eq (6) by ignoring the binary constraints, and then quantize the real-valued latent factors to generate hash codes for users and items. It is worth noting that balanced and decorrelated constraints are also imposed, which can lead to a smaller quantization error [17].

The relaxed problem of Eq (6) is also optimized by alternating optimization. The optimized loss function is the same as Eq (7) for squared loss and as Eq (8) for logistic loss. The updating equations are the same except $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ because of ignorance of binary constraints. Due to advantages of efficiency and effectiveness, coordinate descent is used for optimization, and the updating equations can be derived according to [20], [38], but not elaborated any more in this extension. Its time complexity is $\mathcal{O}(|\Omega|k + (m+n)k^2)$.

Let the solution of the relaxed problem of Eq (6) be $(\hat{\boldsymbol{\Phi}}^*, \hat{\boldsymbol{\Psi}}^*)$, then $\boldsymbol{\Phi}, \boldsymbol{\Psi}$ is initialized to feasible solution

sign($\hat{\mathbf{\Phi}}^*$) and sign($\hat{\mathbf{\Psi}}^*$) respectively, where sign : $R \rightarrow \{\pm 1\}$ is a sign function. Alternatively, ITQ can be used to obtain a better solution, as suggested in [15].

4.4 Extension with Side Information

Users and items usually have auxiliary information, which can significantly improve recommendation performance, particularly in cold-start cases. The modeling capacity of each binary dimension is finite and so much smaller than a real dimension, so many real-valued models for auxiliary information may be not suitable for binary ones. In our preliminary paper, a regression-based model was proposed, to propagate information from auxiliary information to binary codes. In particular, if item's auxiliary information is available, the following constraint is added into Eq (6),

$$\mathcal{R}_4 = \sum_j \|\boldsymbol{\psi_j} - \boldsymbol{U}^T \boldsymbol{y}_j\|^2, \tag{20}$$

where $\boldsymbol{y}_j \in \mathbb{R}^f$ is a feature vector of an item j and $\boldsymbol{U}$ is a regression coefficient, or called a mapping matrix, from $\boldsymbol{y}_j$ to $\boldsymbol{\psi}_j$. Due to data heterogeneity and limited modeling capacity, it is difficult for $\boldsymbol{\psi_j}$ to sufficiently encode auxiliary information of items. Therefore, this method may not work very well. This is also verified in the experiments. Moreover, the coefficient for the constraint needs to be fine-tuned in the validation set. Motivated by collaborative topic regression [39], we propose a novel parameter-free model, called DMF-AUX, for incorporating auxiliary information. Due to the symmetry between user and item, below we only consider how to incorporate items' auxiliary information. Particularly, in addition to binary code $\boldsymbol{\psi}_j$, each item is also encoded with $\boldsymbol{q}_j$ due to its auxiliary information. The final representation is then $\boldsymbol{\psi}_j + \boldsymbol{q}_j$ according to [39]. This is in contrast to the regression-based model only with item code $\boldsymbol{\psi}_j$. In this case, the preference estimation $\hat{r}_{ij}$ can still be efficiently computed based on the following equation,

$$\hat{r}_{ij} = \boldsymbol{\phi}_i^T(\boldsymbol{\psi}_j + \boldsymbol{q}_j) = [\boldsymbol{\phi}_i^T, \boldsymbol{\phi}_i^T] \begin{bmatrix} \boldsymbol{\psi}_j \\ \boldsymbol{q}_j \end{bmatrix}. \tag{21}$$

Since $\boldsymbol{\psi}_j + \boldsymbol{q}_j \in \{-2, 0, +2\}^k$, is not binary any more, we can not use the parameter learning of CTR [39], which learns $\boldsymbol{v}_j = \boldsymbol{\psi}_j + \boldsymbol{q}_j$ and $\boldsymbol{q}_j$ alternatively. To this end, we propose the following objective function in the DMF-AUX model[3] to learn $\boldsymbol{\psi}_j$ and $\boldsymbol{q}_j$ separately,

$$\begin{aligned} \mathcal{L} = & \lim_{\epsilon \rightarrow \infty} \sum_{(i,j) \in \Omega} \Big(r_{ij} - \frac{1}{2}\big(\boldsymbol{\phi}_i^T(\boldsymbol{\psi}_j + \boldsymbol{q}_j)\big)\Big)^2 \\ & + \rho \sum_{(i,j) \notin \Omega} \big(\frac{1}{2}\boldsymbol{\phi}_i^T(\boldsymbol{\psi}_j + \boldsymbol{q}_j)\big)^2 + \epsilon \sum_j \|\boldsymbol{W}\boldsymbol{q}_j - \boldsymbol{y}_j\|^2 \\ & \text{s.t. } \boldsymbol{\phi}_i, \boldsymbol{\psi}_j, \boldsymbol{q}_j \in \{\pm 1\}^k \text{ and } \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k \end{aligned} \tag{22}$$

where ϵ approaching ∞ indicates that the model is hyperparameter-free and that $\boldsymbol{q}_j$ is only determined by its auxiliary information. The coefficient $\frac{1}{2}$ keeps the preference estimation $\hat{r}_{ij}$ in the same range as before. The last term is motivated by SVD of the feature matrix $\boldsymbol{Y}$. Without binary constraints, the optimal $\boldsymbol{W}^\star$ and $\boldsymbol{Q}^\star$ can be yielded by SVD [40]. Essentially, the learning of $\boldsymbol{q}_j$ seamlessly integrates SVD with a well-known hashing methods – ITQ [41]. The column-orthogonal $\boldsymbol{W}$ plays roles in reducing the dimension and balancing the variance.

Its optimization is not identical to Eq (3), but the updating rule of $\boldsymbol{\phi}_i$ is quite similar as long as introducing $\tilde{\boldsymbol{\psi}}_j = \frac{1}{2}(\boldsymbol{\psi}_j + \boldsymbol{q}_j)$. Below we elaborate to derive the updating rule of $\boldsymbol{\psi}_j$. First we only keep the terms dependent on $\boldsymbol{\psi}_j$,

$$\begin{aligned} \mathcal{L}_j = \sum_{i \in \mathbb{U}_j} & \big(\frac{1}{4}\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j - (r_{ij} - \frac{1}{2}\boldsymbol{\phi}_i^T\boldsymbol{q}_j)\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j\big) + \\ & \rho \sum_{i \notin \mathbb{U}_j} \big(\frac{1}{4}\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j + \frac{1}{2}\boldsymbol{\phi}_i^T\boldsymbol{\psi}_j\boldsymbol{\phi}_i^T\boldsymbol{q}_j\big) \end{aligned}$$

This is equivalent to

$$4\mathcal{L}_j = \boldsymbol{\psi}_j^T\boldsymbol{H}_j\boldsymbol{\psi}_j - 2\boldsymbol{\psi}_j^T(2\boldsymbol{\Phi}_j^T\boldsymbol{r}_j - \boldsymbol{H}_j\boldsymbol{q}_j), \tag{23}$$

where $\boldsymbol{H}_j = (1-\rho)\boldsymbol{\Phi}_j^T\boldsymbol{\Phi}_j + \rho\boldsymbol{\Phi}^T\boldsymbol{\Phi}$. Therefore, minimizing this objective function can also be solved by binary quadratic programming using aforementioned algorithms.

Since $\epsilon \rightarrow \infty$, the learning of $\boldsymbol{Q}$ and $\boldsymbol{W}$ is independent to the rating data, and particularly rewritten as follows:

$$\min_{\boldsymbol{W} \in \mathbb{R}^{f \times k}, \boldsymbol{Q} \in \{\pm 1\}^{n \times k}} \|\boldsymbol{Y} - \boldsymbol{Q}\boldsymbol{W}^T\|_F^2, \text{ s.t. } \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k \tag{24}$$

The optimization is prior to the learning of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, and achieved by taking turns updating $\boldsymbol{Q}$ and $\boldsymbol{W}$ until convergence.

Given $\boldsymbol{W}$ fixed, each column of $\boldsymbol{Q}$ can be updated independently to each other. Particularly, the optimal d-th column $\boldsymbol{q}_{(d)}$ is only determined by

$$\min_{\boldsymbol{q}_{(d)} \in \{\pm 1\}^n} \|\boldsymbol{q}_{(d)} - \boldsymbol{Y}\boldsymbol{w}_{(d)}\|^2, \text{ s.t. } \boldsymbol{1}^T\boldsymbol{q}_{(d)} = 0, \tag{25}$$

where $\boldsymbol{w}_{(d)}$ is the d-th column of $\boldsymbol{W}$. Here the balanced constraint $\boldsymbol{1}^T\boldsymbol{Q}$ is additionally imposed, to ensure each bit be as informative as possible. In this case, the optimum $\boldsymbol{q}^\star_{(d)} = \text{sign}(\boldsymbol{Y}\boldsymbol{w}_{(d)} - \text{median}(\boldsymbol{Y}\boldsymbol{w}_{(d)}))$.

Given $\boldsymbol{Q}$ fixed, the objective function with respect to $\boldsymbol{W}$ can be reformulated as follows,

$$\begin{aligned} \min_{\boldsymbol{W}} \|\boldsymbol{Q}\boldsymbol{W}^T - \boldsymbol{Y}\|_F^2 = \|\boldsymbol{W} - \boldsymbol{Y}^T\boldsymbol{Q}\|_F^2 + \text{const}, \\ \text{s.t. } \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k \end{aligned} \tag{26}$$

which is the same as projection of $\boldsymbol{Y}^T\boldsymbol{Q}$ onto the Stiefel manifold as discussed before.

4.5 Complexity Analysis

The overall optimization procedure with detailed time complexity is shown in Algorithm 3. In the algorithm, we first initialize $\boldsymbol{\Phi}, \boldsymbol{\Psi}$ via the initialization algorithm of DMF as discussed in Section 4.3 (Line 1-2) and $\boldsymbol{W}$ via the k right singular vectors of $\boldsymbol{Y}$ (Line 3-4). In the iteration, we first update user binary codes (Line 6-11) and then item binary codes (Line 12-16) via block coordinate descent. In this article, $\boldsymbol{Y}$ is assumed a sparse matrix, so the major cost of partial decomposition based on Krylov subspace methods is from a matrix-vector multiplication of $\boldsymbol{Y}$. This cost is denoted T_{mult}. The cost of learning $\boldsymbol{Q}$ is $\mathcal{O}(kT_{mult} + (n+f)k^2)$ [42]. The overall cost of learning $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ is $\mathcal{O}((m+n)k_1^{2.5}k\#iter + |\Omega|k^2)$, where k_1 is the block size and $\#iter$ is the number of iterations in BCD until convergence. In practice, $\#iter = 1$ works well.

3. For brevity, we ignore the balanced and decorrelation constraints and focus on the squared loss function.

Algorithm 3: DMF-AUX

Input: Rating matrix $\boldsymbol{R}$, item feature $\boldsymbol{Y}$, bit length k, regularization coefficient ρ.
Output: $\boldsymbol{\Phi}, \boldsymbol{\Psi}, \boldsymbol{Q}$

$\hat{\boldsymbol{\Phi}}^*, \hat{\boldsymbol{\Psi}}^* \leftarrow$ DMF_i($\boldsymbol{R}$, k, ρ); // $\mathcal{O}(|\Omega|k + (m+n)k^2)$
$\boldsymbol{\Phi}, \boldsymbol{\Psi} \leftarrow \text{sign}(\hat{\boldsymbol{\Phi}}^*), \text{sign}(\hat{\boldsymbol{\Psi}}^*)$;
$\sim, \sim, \boldsymbol{W} \leftarrow$ SVD($\boldsymbol{Y}, k$); // $\mathcal{O}(kT_{mult} + (f+n)k^2)$
repeat
  for $d = \{1, \cdots, k\}$ **do**
    $\boldsymbol{q}_{(d)} \leftarrow$ Solve Eq (25); // $\mathcal{O}(T_{mult})$
  $\boldsymbol{W} \leftarrow$ Solve Eq (26); // $\mathcal{O}(kT_{mult} + fk^2)$
until *Convergent*;
repeat
  $\tilde{\boldsymbol{\Psi}} = \frac{1}{2}(\boldsymbol{\Psi} + \boldsymbol{Q})$;
  Prepcompute $\tilde{\boldsymbol{\Psi}}^T\tilde{\boldsymbol{\Psi}}$; // $\mathcal{O}(nk^2)$
  for $i = \{1, \cdots, m\}$ **do**
    $\boldsymbol{A}_i \leftarrow (1-\rho)\tilde{\boldsymbol{\Psi}}_i^T\tilde{\boldsymbol{\Psi}}_i + \rho\tilde{\boldsymbol{\Psi}}^T\tilde{\boldsymbol{\Psi}}$; // $\mathcal{O}(|\mathbb{I}_i|k^2)$
    $\boldsymbol{b}_i \leftarrow \tilde{\boldsymbol{\Psi}}_i^T\boldsymbol{r}_i$; // $\mathcal{O}(|\mathbb{I}_i|k)$
    $\boldsymbol{\phi}_i \leftarrow$ BCD($\boldsymbol{A}_i, \boldsymbol{b}_i$); // $\mathcal{O}(k_1^{2.5}k\#iter)$
  Precompute $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$; // $\mathcal{O}(mk^2)$
  for $j = \{1, \cdots, n\}$ **do**
    $\boldsymbol{H}_j \leftarrow (1-\rho)\boldsymbol{\Phi}_j^T\boldsymbol{\Phi}_j + \rho\boldsymbol{\Phi}^T\boldsymbol{\Phi}$; // $\mathcal{O}(|\mathbb{U}_j|k^2)$
    $\boldsymbol{g}_j \leftarrow 2\boldsymbol{\Phi}_j^T\boldsymbol{r}_j - \boldsymbol{H}_j\boldsymbol{q}_j$; // $\mathcal{O}(|\mathbb{U}_j|k + k^2)$
    $\boldsymbol{\psi}_j \leftarrow$ BCD($\boldsymbol{H}_i, \boldsymbol{g}_i$); // $\mathcal{O}(k_1^{2.5}k\#iter)$
until *Convergent*;

5 Two-Stage Recommender System

Several hashing-based recommendation algorithms were proposed [15], [16], [17], but its practical value for real-world recommender systems is not sufficiently discussed. The knowledge these literatures deliver is that these algorithms can improve the efficiency at the expense of recommendation performance. However, if the recommendation performance degrades a lot, these algorithms may be not useful in practice. Deviation from evaluation of real-world recommender systems makes the reported results about small recommendation performance degradation untrusted. Therefore, we propose a two-stage recommender system, consisting of a hashing-based item-recalling stage and a fine-ranking stage. The framework is demonstrated in Fig. 3.

In the first stage, hashing-based recommendation algorithms will be trained on the input rating matrix, the auxiliary information of users and items. These algorithms will output user binary codes and item binary codes. Given each user's binary code, we can estimate his coarse preference for all unrated items based on the extract/approximated top-K nearest neighbor (NN) search. Since preferences are estimated as hamming distance between binary codes, its computation can be very efficient via bit-wise operations. And preferences are integer-valued and bounded, so the full ranking of all unrated items with respect to each user is also very efficient. Furthermore, the extract knn search can be accelerated by multi-index hashing [10], whose time complexity can be sublinear. The returned items by the item-recalling stage are considered item candidate, in which the user may be interested. Note that the number of candidate items is much smaller than unrated items.

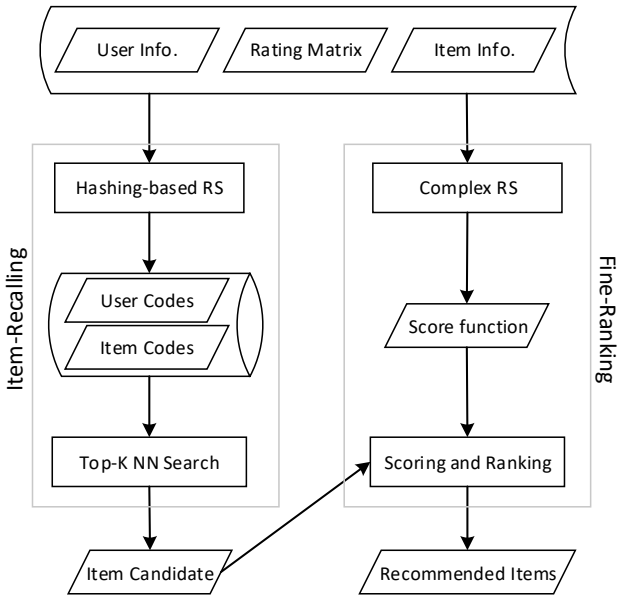


Fig. 3. The framework of the two-stage recommender system

In the second stage, many complex recommendation algorithms will be also trained on input rating matrix, the auxiliary information of users and items. Currently, we assume the second stage algorithms are trained independently to the first stage algorithms. Guiding the training of the second-stage model with the first-stage model is left for future work. These complex recommendation can be Factorization Machine [43], Collaborative Topic Regression [39], Neural Factorization Machine [44], Collaborative Knowledge Embedding [45] and Variational Autoencoders [46]. In these models, user parameters are usually interacted with items' and even dot product will be replaced some non-linear functions, so the estimation of preference may be highly time-consuming. Therefore, it is often the case that only a small part of randomly-drawn items are considered as item candidate [44]. This is not very reasonable from the perspectives of real-world recommender systems. Instead, it is much more reasonable that these complex algorithms, outputting a scoring function after training, score and rank the candidate items returned by the item-recall stage. Since the number of candidate items is much smaller than unrated items, time cost of this step has been dramatically reduced.

Therefore, the proposed two-stage recommender system also addresses the efficiency issues of recommendation of complex models, at the same time of better understanding the benefit of hash-based recommender systems in real-world scenarios. This plays an important role in incorporating more complex recommendation algorithms into practical recommender systems.

6 Experiments

6.1 Datasets

The evaluation of discrete matrix factorization is conducted on 4 explicit feedback datasets and 4 implicit feedback datasets. The 4 explicit feedback datasets are also converted into implicit feedback by treating ratings of higher than each

user's average score as "like" preference. Table 1 summarizes statistics of these datasets. The datasets vary in the numbers of items and ratings. The Yelp dataset includes users' ratings for points of interest. The Amazon dataset is a subset of customers' ratings for Amazon books [47]. The Netflix dataset is from the well-known Netflix Prize. The rating scores of these three datasets are integers from 1 to 5. The MovieLens dataset is from the classic MovieLens10M dataset. The rating scores are from 0.5 to 5 with 0.5 granularity. Following convention of evaluating CF algorithms, we filter these 4 datasets such that users rated at least 20 items that were rated by at least 20 users. The first implicit dataset is CiteULike [48], where articles are collected by users into their reference libraries. The Gowalla dataset includes users' check-ins at locations. Because of low density, it is less strictly filtered such that users check-in at least 10 locations which were checked in by 10 users. The LastFM and EchoNest dataset are based on users' play count of songs. Following [49], we include songs a user listened to at least 5 times as positive feedback.

When evaluating the extension of discrete matrix factorization, we only choose the Amazon dataset and the Yelp dataset, where most items are associated to a set of textual reviews. For each item, we aggregate all textual reviews, filter stop words and represent them by bag of words. We follow [39] to use tf-idf for picking up the top 8,000 distinct words into the vocabulary. To better show the effect of auxiliary information, we only remove users and items with fewer than 10 items. Table 2 summaries data statistics.

For each user, we randomly sample his 80% ratings as training set and the rest 20% as testing test. We fit a model to the training set and evaluate it in the test set. We repeat 5 random splits and report the averaged recommendation performance metrics.

6.2 Evaluation Metrics

The recommendation performance is assessed by how well rated items in the test set are ranked among all unrated items in the training set. We exploit three widely-used metrics in ranking evaluation: Area under ROC curve (AUC) [50], Recall [28], [39] and NDCG [18]. AUC measures the probability that a randomly "liked" (i.e. rating larger than user's average score) items in the test set ranks above a randomly chosen "disliked" items. Recall at cutoff K, denoted as Recall@K, is the fraction of "liked" items in the top-K ranking list over the total number of "liked" items in the test set. NDCG at cutoff K, denoted as NDCG@K, rewards method that ranks "liked" items at the top of the top-K ranking list. The "liked" items ranked at low positions of ranking list contribute less than "liked" items at top positions. In contrast to NDCG, no discount factors are used in AUC and Recall so that items in the ranking list are treated equally. AUC and Recall differ in the length of ranking list, and AUC consider the ranking list of all unrated items in the training set.

We also report the NDCG computed only on the rated items in the testing set. This metric was used in previous studies on binary representation of recommender systems [15], [16], [17], but does not consider the real world case scenario in which all unrated items in the training set should be ranked [29].

6.3 Parameter Settings

The code length of binary representation is set 64 by default. The parameters (ρ, α, β) are tuned by held-out validation, i.e., 10% of the training split. ρ ($\leq$ 1) is tuned within $\{10^{-6}, 10^{-5}, \cdots, 10^{-1}, 1\}$, both α and β are tuned within $\{10^{-4}, 10^{-3}, \cdots, 10^{1}, 10^{2}\}$. The number of iteration is set to 20. The algorithms are implemented via MATLAB with MEX C++ and released in a open-source framework of a MATLAB-based recommender system[4].

6.4 Baselines of Comparison

We compare discrete matrix factorization with the following baselines.

- **DCF** [17], the state-of-the-art regression-based method without auxiliary information and also directly tackles a discrete optimization problem via cyclic coordinate descent, subject to the decorrelated and balanced constraints. The parameters α and β for the decorrelated and balanced constraints are tuned within $\{10^{-4}, 10^{-3}, \cdots, 10^{1}, 10^{2}\}$.
- **BCCF** [15], is a binary code learning method for collaborative filtering. It solves a relaxed matrix factorization but imposes a balanced regularization for latent factors instead of the ℓ_2-norm regularization. It then uses the ITQ method [41] to derive the binary codes for users and items. The coefficient for the balanced regularization is tuned within $\{0.01, 0.03, 0.05, 0.07, 0.09\}$ according to the results of their sensitive analysis.
- **PPH** [16], is a preference preserving hashing based matrix factorization. Different from BCCF, PPH imposes constant feature norm constraints so that users' preferences can be well approximated by similarities, since the authors argued that quantization would loss the magnitude information of latent factors. PPH then quantizes each latent factor into k-bit phrase codes and 2-bit magnitude codes. The coefficient for the constant feature norm is tuned within $\{0.01, 0.5, 1, 2, 4, 8, 16\}$.
- **CH** [14], Collaborative Hashing is also a heuristic method for learning binary codes. CH first solves matrix factorization on the full-matrix, by treating all unrated items as zero-rated. Following [17], we also implement CH as $\arg\min_{\boldsymbol{U},\boldsymbol{V}} \|\boldsymbol{R} - \boldsymbol{U}\boldsymbol{V}^T\|_F^2$, s.t. $\boldsymbol{U}^T\boldsymbol{U} = m\boldsymbol{I}_k, \boldsymbol{V}^T\boldsymbol{V} = n\boldsymbol{I}_k$. CH then quantizes the $\boldsymbol{U}$ and $\boldsymbol{V}$ based on the sign function.

The comparison with real-valued matrix factorization will be studied when evaluating the two-stage recommender system. The source codes of these baselines and the two-stage recommender system are also released in the github repository.

We then compare the extension of discrete matrix factorization for auxiliary information, denoted as **DMF-AUX**, with the following baselines.

- **DCMF**, the extension of DMF in our preliminary work [20] based on the regression-based modeling. In this work, we assumes binary codes are determined by ratings and auxiliary information jointly. The parameter λ_2 for modeling item textual features is tuned within $\{0, 1, 10, 50, 100, 500, 1000\}$. It is worth noting that this parameter in the DCMF initialization algorithm should be

4. https://github.com/DefuLian/recsys

TABLE 1
Statistics of 8 datasets for evaluating discrete matrix factorization.

Datasets	Yelp	Amazon	MovieLens	Netflix	CiteULike	Gowalla	Lastfm	EchoNest
#users	18,454	35,736	69,838	429,584	7,947	29,858	357,847	766,882
#items	14,670	38,121	8,939	17,764	25,975	40,988	156,122	260,417
#ratings	869,126	1,960,674	9,983,739	99,884,887	134,860	1,027,464	16,893,651	7,261,443
Density	3.21e-03	1.44e-03	1.60e-02	1.31e-02	6.53e-04	8.40e-04	3.02e-04	3.64e-05

TABLE 2
Statistics of the Yelp and Amazon datasets for evaluating the extension of discrete matrix factorization.

Datasets	#users	#items	#ratings	Density	Feature size
Yelp	77,277	45,638	2,103,895	0.06%	51,056,602
Amazon	158,650	128,939	4,701,968	0.02%	127,755,615

re-tuned, since loss value in the rating-based part may change a lot from real-valued latent factors to binary codes.

- **DMF+DH**, this is a straightforward baseline, which learns item Document Hash (DH) codes and binary codes in DMF independently. Then each item is represented by direct addition between document hash code and item binary code. The difference from DMF-AUX lies in the optimization algorithm.
- **DMF**, without auxiliary information considered.

Also, the comparison with a real-valued method, i.e., collaborative topic regression [39] will be studied when evaluating the two-stage recommender system at the presence of auxiliary information of items.

6.5 Results and Analysis

6.5.1 Comparison with the State of the Art

Table 3 shows the recommendation performance, including NDCG@100, Recall@100, and AUC, as well as NDCG-RI (NDCG on the rated items only) of DMF and competing baselines on explicit feedback. We have the following key observations.

First, the proposed DMF algorithm consistently and significantly outperforms the state of the art with respect to Recall@100 and NDCG@100. The improvements in the denser datasets such as MovieLens and Netflix are higher than 300%. DMF is also significantly better than the state of the art with respect to AUC in all datasets except Yelp. In contrast, though DCF shows better recommendation performance with respect to NDCG-RI, the differences from other algorithms are marginal. Therefore, biasing the objective function with the interaction regularization may loose performance of ranking rated items only but result in the superior performance of DMF of ranking all unrated items in the training set to baselines. Of course, the superior performance of DMF is also induced by the balanced and decorrelated constrains.

Second, CH shows good recommendation performance compared to BCCF and PPH, and sometimes even better than DCF. One important potential reason is the consideration of all unrated items as zero-rated. This is similar to DMF by setting $\rho = 1$, meaning that the interaction regularization plays the same important role in the rating-regression loss. Another possible reason is that CH incorporates the decorrelated constraints for learning relaxed latent factors of users and items.

Table 4 shows the recommendation performance including NDCG@100, Recall@100 and AUC of DMF and competing baselines in the 8 implicit feedback datasets. Note that there is no rating in implicit feedback, so that NDCG-RI can not be computed any more. These results can not be compared against that of explicit feedback, since the testing set is not of the same size. DMF has two versions, denoted as DMF-l with logistic loss and DMF-s with squared loss, respectively. From this table, we observe that DMF-s outperforms the state of the art significantly with respect to NDCG@100 and Recall@100 in all 8 datasets. However, DMF-l only shows better performance than baselines in some datasets, such as Movielens, Netflix and LastFM. This, on one hand, shows the effectiveness of the proposed algorithm based on Majorization-Minimization, on the other hand, reveals that DMF-l easily suffers from sparsity issues. This also implies the difficulty of learning binary representation in case of non-squared loss functions. The exploration of non-squared loss function will be reserved for future work. Similar to results in explicit feedback, CH still shows good performance, mainly induced by the consideration of all items in the loss function instead of only "liked" items.

6.5.2 Sensitivity Analysis - I

After illustrating the superior recommendation performance of DMF, we then conduct sensitivity analysis with respect to optimization algorithms, code length k, coefficients α and β for balanced and decorrelated constraints respectively. The results of optimization algorithms are shown in Fig. 4. We can see that on three denser datasets, the BCD-based optimization algorithm shows better recommendation performance with respect to Recall[5]. The improvements with respect to Recall@100 are up to 9.8%, 5.1%, 7.7% in the Yelp, Movielens, Netflix dataset respectively. With the increase of coordinate block size, the loss function is reduced more and more, as shown in Fig. 1 and Fig. 2. Recall becomes better in the Yelp dataset, but worse in the MovieLens and Netflix datasets. This is mainly because the loss function is not based on the ranking metric – Recall. However, this implies that we can choose a smaller size of coordinate blocks for optimization and a lower-precision SDR solver for BQP. The other results of sensitivity analysis are shown Fig. 5. With the increase of code length from 16 to 256, the recommendation performance gradually improves in all of the four datasets and more rapidly in the sparser datasets.

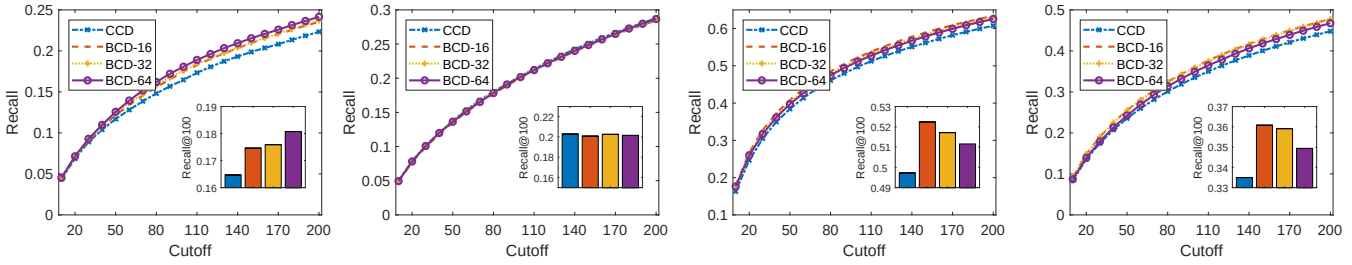
5. Only Recall is reported since the retrieval of the top-K potentially preferred items is the most concerned.

TABLE 3
Comparison with the State of the Art on Explicit Feedback Datasets.

	NDCG-RI	NDCG@100	Recall@100	AUC	DDCG-RI	NDCG@100	Recall@100	AUC
	Yelp				Amazon			
PPH	0.9456±0.0004	0.0117±0.0005	0.0328±0.0010	0.6751±0.0023	0.9671±0.0002	0.0088±0.0002	0.0218±0.0004	0.6949±0.0011
BCCF	0.9472±0.0003	0.0265±0.0003	0.0757±0.0010	**0.8541**±0.0008	0.9630±0.0002	0.0468±0.0003	0.1015±0.0005	0.8139±0.0007
DCF	**0.9484**±0.0003	0.0220±0.0004	0.0635±0.0015	0.7725±0.0046	**0.9686**±0.0001	0.0216±0.0005	0.0518±0.0015	0.8248±0.0020
CH	0.9304±0.0003	0.0484±0.0011	0.1320±0.0023	0.7737±0.0015	0.9590±0.0002	0.0632±0.0005	0.1394±0.0011	0.8339±0.0010
DMF	0.9423±0.0002	**0.0733**±0.0007	**0.1656**±0.0021	0.7375±0.0011	0.9673±0.0002	**0.0920**±0.0006	**0.2015**±0.0012	**0.8850**±0.0004
	MovieLens				Netflix			
PPH	0.9534±0.0002	0.0367±0.0004	0.0735±0.0008	0.7409±0.0019	0.9529±0.0001	0.0235±0.0001	0.0397±0.0003	0.7190±0.0011
BCCF	0.9475±0.0002	0.0725±0.0002	0.1169±0.0002	0.7476±0.0003	0.9473±0.0001	0.0564±0.0001	0.0752±0.0002	0.7003±0.0002
DCF	**0.9620**±0.0001	0.0944±0.0011	0.1780±0.0010	0.8237±0.0005	**0.9630**±0.0000	0.0661±0.0006	0.1093±0.0008	0.8225±0.0005
CH	0.9302±0.0001	0.0854±0.0012	0.1463±0.0017	0.6836±0.0013	0.9378±0.0002	0.0729±0.0013	0.1011±0.0020	0.6724±0.0017
DMF	0.9576±0.0001	**0.2999**±0.0011	**0.4984**±0.0005	**0.9025**±0.0005	0.9571±0.0000	**0.2274**±0.0004	**0.3339**±0.0007	**0.8562**±0.0003

TABLE 4
Comparison with the State of the Art on Implicit Feedback Datasets.

	NDCG@100	Recall@100	NDCG@100	Recall@100	NDCG@100	Recall@100	NDCG@100	Recall@100
	Yelp		Amazon		MovieLens		Netflix	
PPH	0.0197±0.0011	0.0558±0.0024	0.0133±0.0007	0.0342±0.0019	0.0412±0.0058	0.0940±0.0118	0.0345±0.0076	0.0569±0.0119
BCCF	0.0316±0.0002	0.0885±0.0014	0.0461±0.0008	0.1021±0.0013	0.0520±0.0002	0.0857±0.0004	0.0290±0.0000	0.0370±0.0001
DCF	0.0558±0.0006	0.1577±0.0017	0.0588±0.0003	0.1385±0.0005	0.0545±0.0013	0.1154±0.0026	0.0046±0.0001	0.0083±0.0001
CH	0.0353±0.0005	0.0969±0.0017	0.0528±0.0005	0.1160±0.0009	0.0652±0.0010	0.1154±0.0025	0.0620±0.0012	0.0866±0.0017
DMF-l	0.0524±0.0006	0.1299±0.0013	0.0522±0.0002	0.1187±0.0005	0.2136±0.0008	0.3807±0.0016	0.1631±0.0001	0.2583±0.0002
DMF-s	**0.0930**±0.0010	**0.2467**±0.0021	**0.1004**±0.0006	**0.2255**±0.0010	**0.2840**±0.0011	**0.5203**±0.0014	**0.2265**±0.0016	**0.3629**±0.0029
	CiteULike		Gowalla		LastFM		EchoNest	
PPH	0.0514±0.0013	0.1350±0.0023	0.0201±0.0012	0.0536±0.0031	0.0167±0.0010	0.0362±0.0027	0.0051±0.0005	0.0169±0.0024
BCCF	0.0570±0.0009	0.1559±0.0022	0.0834±0.0008	0.1968±0.0015	0.0464±0.0001	0.0868±0.0002	0.0333±0.0001	0.0774±0.0002
DCF	0.0707±0.0015	0.1854±0.0029	0.0898±0.0016	0.2234±0.0036	0.0434±0.0028	0.0895±0.0055	0.0159±0.0005	0.0462±0.0013
CH	0.0345±0.0011	0.1032±0.0022	0.0563±0.0008	0.1365±0.0019	0.0257±0.0002	0.0506±0.0003	0.0074±0.0003	0.0224±0.0008
DMF-l	0.0473±0.0019	0.1171±0.0029	0.0983±0.0012	0.2064±0.0016	0.1403±0.0003	0.2483±0.0005	0.0386±0.0002	0.0977±0.0003
DMF-s	**0.1144**±0.0016	**0.2857**±0.0027	**0.1390**±0.0012	**0.3049**±0.0015	**0.1826**±0.0006	**0.3228**±0.0008	**0.0788**±0.0006	**0.2102**±0.0004



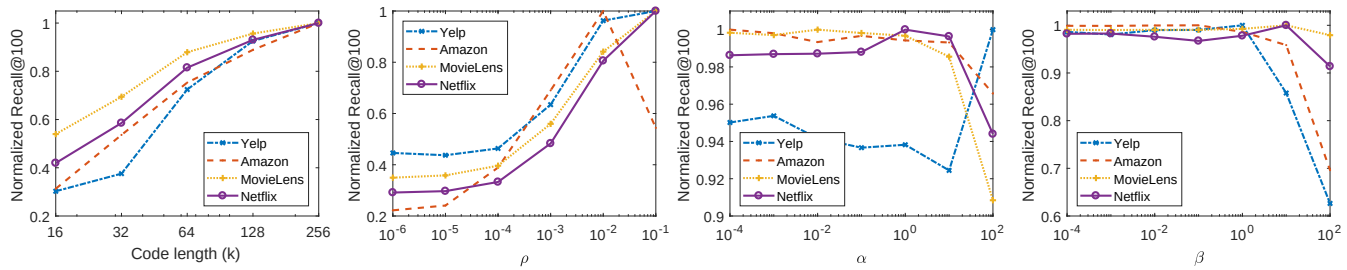
(a) Yelp (b) Amazon (c) MovieLens (d) Netflix

Fig. 4. Sensitive analysis of the DMF model in the four datasets.

The importance of interaction regularization can be shown again in Fig. 5(b). This figure also shows that $\rho \in [10^{-3}, 1)$ can lead to quite good recommendation performance and ρ should be set to a smaller value in sparser datasets. According to Fig. 5(c), we observe that balanced regularization can take effect at deriving more informative binary codes in the Yelp and Netflix datasets. And in the Yelp dataset, the improvements of recommendation performance can be up to 5%. However, the effect of the decorrelation constraint looks marginal, except in the Netflix dataset. This doesn't mean that the decorrelation constraint is not useful for deriving the compact binary codes, but implies such a method may be not effective for very sparse data, such as rating/preference matrix in recommender systems. This direction will also be reserved for future work.

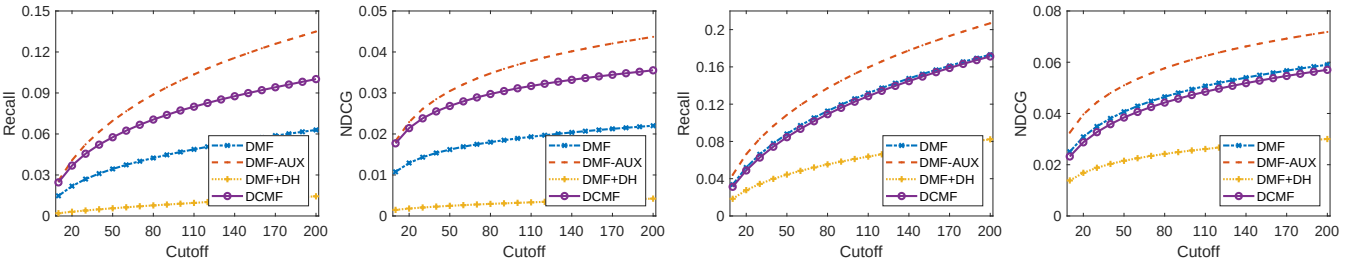
6.5.3 *Evaluating the Effect of Auxiliary Information*

After studying DMF, we are then concerned with the effect of auxiliary information. The results are shown in Fig. 6. It is easily observed that the proposed DMF-AUX consistently outperforms DMF in the both datasets, showing the effectiveness of DMF-AUX for modeling auxiliary information. However, DCMF only shows superior performance in the Yelp dataset to DMF, revealing the problems of DCMF for modeling auxiliary information in case of data heterogeneity and limited modeling capacity. Note that DCMF is proposed



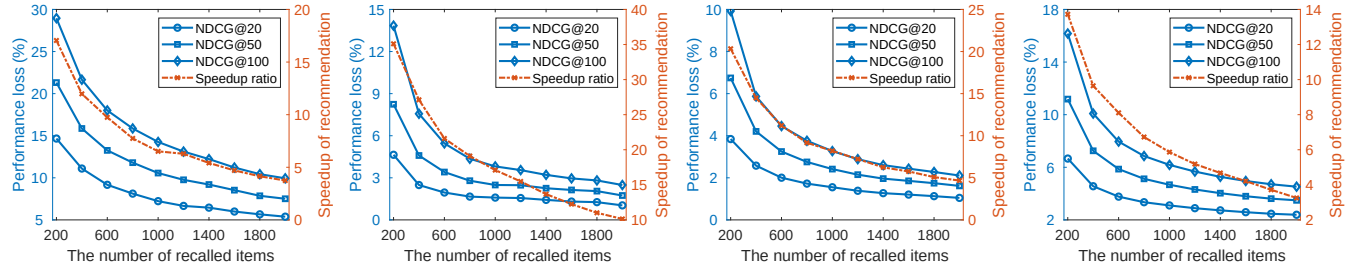
(a) Code length (b) Regularization (c) Balance (d) Decorrelation

Fig. 5. Sensitive analysis of the DMF model in the four datasets, where Normalized NDCG@100 is obtained by dividing each NDCG@100 by the maximum with respect to the parameter.



(a) Yelp-Recall (b) Yelp-NDCG (c) Amazon-Recall (d) Amazon-NDCG

Fig. 6. Results of evaluating the extension of DMF for auxiliary information in the Yelp and Amazon datasets.



(a) Yelp (b) Amazon (c) MovieLens (d) Netflix

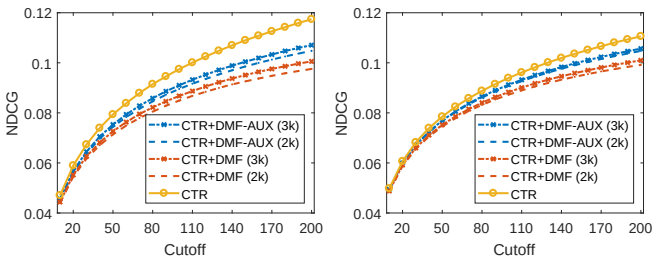
Fig. 7. Results of evaluating the two-stage model in the four datasets.

based on feature-aware matrix factorization [51], [52], so that the methods being appropriate for real-valued algorithms may not work well for binary-valued ones. Comparing DMF-AUX with DCMF, DMF-AUX shows much better performance than DCMF in the Amazon dataset and the Yelp dataset. The straightforward baseline, DMF+DH, is even much worse than DMF, indicating simple aggregation of multi-modal hash codes can not work well.

6.5.4 *Evaluating the Two-Stage Recommender Systems*

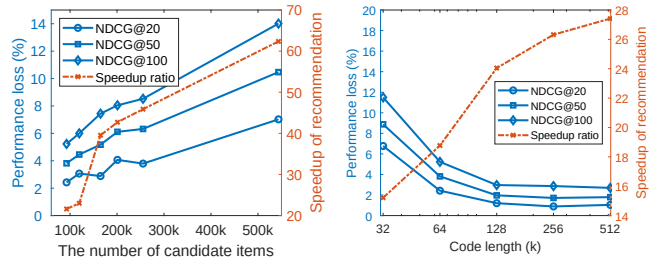
Evaluating the two-stage recommender system can help us better understand how DMF can accelerate practical recommender systems. In this evaluation, the first stage is to exploit DMF for retrieving the top-K potentially preferred items, and the second stage is to use real-valued MF with interaction regularization [28], [29], [30] for fine-ranking. Such a choice of the fine-ranking algorithm lies in its simplicity and superiority compared to other more complex algorithms. The results in the four datasets are shown in Fig. 7 when the number of recalled items varies. We can see that when the number of recalled items is 600, the degradation of NDCG@20 is 9.2% in the Yelp dataset and less than 3.8% in the other datasets. There is usually larger degradation of NDCG at the lower cutoffs. In spite of degradation of recommendation performance, the recommendation can be accelerated by more than 8 times, and even up to around 22 in the Amazon dataset with the largest number of items. With the increasing number of recalled items, the degradation of recommendation performance will be smaller and smaller, but the speedup ratio is decreasing. Therefore, we usually need to strike a balance between efficiency and effectiveness of recommender systems. Though the speedup ratio is not so large, the efficiency can be further improved based on fully C++ implementation and more advanced algorithms like multi-index hashing as observed in [16].

When auxiliary information is available, the results of evaluation are shown in Fig. 8, where the recalled items only occupy smaller than 7%. Here collaborative topic regression [39] is placed in the second stage of fine-ranking for recalled items. For understanding the effect of auxiliary information, both DMF and DMF-AUX are placed in the first-stage item-recall algorithm. From this figure, we observe that, similar to Section 6.5.3, auxiliary information can take significant effect in retrieving the better top-K preferred items. This



(a) Yelp (b) Amazon

Fig. 8. Results of evaluating the extension of DMF for auxiliary information in the two-stage recommender system. '2k' and '3k' indicates the number of recalled items, meaning 2000 and 3000 respectively.



(a) Item size (b) Code length

Fig. 9. Sensitive analysis of the two-stage model in the Amazon dataset.

means the performance degradation resulting from quantization can be further reduced by making use of auxiliary information.

6.5.5 Sensitive Analysis — II

We observe that speedup ratio in the datasets of more items is usually larger, but it is unknown how the efficiency and effectiveness of this two-stage recommender system varies with the change of item size and code length. Therefore, we conduct the sensitivity analysis in the two-stage recommender system by placing DMF in the first stage in the Amazon dataset, since the number of its items is much larger and over 1 million without any filtering. The evaluation results are shown in Fig. 9. We can easily observe that with the increasing number of items, the speedup ratio grows larger and larger. When item size is round 500K, item recommendation can be 60+ times faster. However, the recommendation performance degrades more, by up to 7%, 10% and 14% with respect to NDCG@20, NDCG@50 and NDCG@100. This again shows that DMF easily suffers from sparsity issues. With the increase of code length, the performance degradation begins a gradual drop toward a nadir around k=128 and then stays at a very low value. Online item recommendation speeds up more and more with the increase of dimension since the computational cost of dot product grows much faster than that of hamming distance. Therefore, we can strike a balance between efficiency and effectiveness by growing code length in addition to incorporating auxiliary information as discussed in the previous section. Moreover, we believe that discrete matrix factorization and extension can be integrated with other methods, such as popularity-based ranking and content-based filtering, for recalling better potentially preferred items.

7 Conclusions and Future Works

In this paper, we proposed a general framework for discrete matrix factorization and its extension. The framework can optimize multiple loss functions, handle both explicit and implicit feedback datasets, and incorporate auxiliary information without hyperparameters. The evaluation on 8 real-world datasets showed that the proposed algorithms outperformed the state of the art significantly and consistently, and that item auxiliary information dramatically improved the recommendation performance. Moreover, the framework was optimized by a novel block coordinate descent algorithm, so that the loss function was reduced much more and the recommendation performance was significantly improved. DMF was then treated as the first stage of the proposed two-stage recommender system for recalling the top-K potentially preferred items, and shown its advantages for striking a balance between efficiency and effectiveness of practical recommender systems.

Future work includes carefully designing optimization algorithms for non-squared loss functions, investigating how to deal with sparsity issues in recommender systems, and extensively studying more first-stage algorithms and their combination in the two-stage recommender systems.

Acknowledgments

The work was supported in part by grants from the National Natural Science Foundation of China (Grant No. 61976198, U1605251, 61832017 and 61631005).

References

[1] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of WWW'07*. ACM, 2007, pp. 271–280.

[2] B. Wang, M. Ester, J. Bu, Y. Zhu, Z. Guan, and D. Cai, "Which to view: Personalized prioritization for broadcast emails," in *Proceedings of WWW'16*, 2016, pp. 1181–1190.

[3] B. Wang, M. Ester, Y. Liao, J. Bu, Y. Zhu, Z. Guan, and D. Cai, "The million domain challenge: Broadcast email prioritization by cross-domain recommendation," in *Proceedings of KDD'16*. ACM, 2016, pp. 1895–1904.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of WWW'17*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[5] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: modeling user behaviors by time-lstm," in *Proceedings of IJCAI'17*. AAAI Press, 2017, pp. 3602–3608.

[6] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *Proceedings of KDD'18*. ACM, 2018, pp. 1754–1763.

[7] Y. Zhu, J. Zhu, J. Hou, Y. Li, B. Wang, Z. Guan, and D. Cai, "A brand-level ranking system with the customized attention-gru model," in *Proceedings of IJCAI'18*. AAAI Press, 2018, pp. 3947–3953.

[8] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Know. Data. Eng.*, vol. 17, no. 6, pp. 734–749, 2005.

[9] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[10] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast search in hamming space with multi-index hashing," in *Proceedings of CVPR'12*. IEEE, 2012, pp. 3108–3115.

[11] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, 2012.

[12] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proceedings of VISAPP'09*, 2009, pp. 331–340.

[13] J. Håstad, "Some optimal inapproximability results," *Journal of the ACM (JACM)*, vol. 48, no. 4, pp. 798–859, 2001.

[14] X. Liu, J. He, C. Deng, and B. Lang, "Collaborative hashing," in *Proceedings of CVPR'14*, 2014, pp. 2139–2146.

[15] K. Zhou and H. Zha, "Learning binary codes for collaborative filtering," in *Proceedings of KDD'12*. ACM, 2012, pp. 498–506.

[16] Z. Zhang, Q. Wang, L. Ruan, and L. Si, "Preference preserving hashing for efficient recommendation," in *Proceedings of SIGIR'14*. ACM, 2014, pp. 183–192.

[17] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete collaborative filtering," in *Proceedings of SIGIR'16*. ACM, 2016, pp. 325–334.

[18] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola, "Maximum margin matrix factorization for collaborative ranking," *Proceedings of NIPS'07*, pp. 1–8, 2007.

[19] Z.-Q. Luo, W.-K. Ma, A. M.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20–34, 2010.

[20] D. Lian, R. Liu, Y. Ge, K. Zheng, X. Xie, and L. Cao, "Discrete content-aware matrix factorization," in *Proceedings of KDD'17*, 2017, pp. 325–334.

[21] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to hash for indexing big data – a survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.

[22] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *arXiv preprint arXiv:1606.00185*, 2016.

[23] L. Li, D. Wang, T. Li, D. Knox, and B. Padmanabhan, "Scene: a scalable two-stage personalized news recommendation system," in *Proceedings of SIGIR'11*. ACM, 2011, pp. 125–134.

[24] A. Karatzoglou, A. J. Smola, and M. Weimer, "Collaborative filtering on a budget," in *Proceedings of AISTATS'10*, 2010, pp. 389–396.

[25] Y. Zhang, D. Lian, and G. Yang, "Discrete personalized ranking for fast collaborative filtering from implicit feedback," in *Proceedings of AAAI'17*, 2017, pp. 1669–1675.

[26] Y. Zhang, H. Yin, Z. Huang, X. Du, G. Yang, and D. Lian, "Discrete deep learning for fast content-aware recommendation," in *Proceedings of WSDM'18*. ACM, 2018, pp. 717–726.

[27] H. Liu, X. He, F. Feng, L. Nie, R. Liu, and H. Zhang, "Discrete factorization machines for fast feature-based recommendation," in *Proceedings of IJCAI'18*. International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 3449–3455.

[28] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proceedings of ICDM'08*. IEEE, 2008, pp. 263–272.

[29] R. Devooght, N. Kourtellis, and A. Mantrach, "Dynamic matrix factorization with priors on unknown values," in *Proceedings of KDD'15*. ACM, 2015, pp. 189–198.

[30] I. Bayer, X. He, B. Kanagal, and S. Rendle, "A generic coordinate descent framework for learning from implicit feedback," in *Proceedings of WWW'17*. International World Wide Web Conferences Steering Committee, 2017, pp. 1341–1350.

[31] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz, "An interior-point method for semidefinite programming," *SIAM Journal on Optimization*, vol. 6, no. 2, pp. 342–361, 1996.

[32] Y. Nesterov *et al.*, "Semidefinite relaxation and nonconvex quadratic optimization," Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), Tech. Rep., 1997.

[33] N. Koenigstein and U. Paquet, "Xbox movies recommendations: Variational bayes matrix factorization with embedded feature selection," in *Proceedings of RecSys'13*. ACM, 2013, pp. 129–136.

[34] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proceedings of NIPS'09*, 2009, pp. 1753–1760.

[35] T. Jaakkola and M. Jordan, "A variational approach to bayesian logistic regression models and their extensions," in *Sixth International Workshop on Artificial Intelligence and Statistics*, vol. 82, 1997.

[36] L. Eldén and H. Park, "A procrustes problem on the stiefel manifold," *Numerische Mathematik*, vol. 82, no. 4, pp. 599–619, 1999.

[37] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge press, 1990.

[38] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proceedings of SIGIR'16*, vol. 16, 2016.

[39] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of KDD'11*. ACM, 2011, pp. 448–456.

[40] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proceedings of ICML'03*, 2003, pp. 720–727.

[41] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, 2013.

[42] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[43] S. Rendle, "Factorization machines with libfm," *ACM Trans. Intell. Syst. Tech.*, vol. 3, no. 3, p. 57, 2012.

[44] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of SIGIR'17*. ACM, 2017, pp. 355–364.

[45] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of KDD'16*. ACM, 2016, pp. 353–362.

[46] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of WWW'18*. International World Wide Web Conferences Steering Committee, 2018, pp. 689–698.

[47] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of KDD'15*. ACM, 2015, pp. 785–794.

[48] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of KDD'15*. ACM, 2015, pp. 1235–1244.

[49] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin, "Collaborative metric learning," in *Proceedings of WWW'17*. International World Wide Web Conferences Steering Committee, 2017, pp. 193–201.

[50] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of UAI'09*. AUAI Press, 2009, pp. 452–461.

[51] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *Proceedings of KDD'09*. ACM, 2009, pp. 19–28.

[52] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu, "Svdfeature: a toolkit for feature-based collaborative filtering," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3619–3622, 2012.

Defu Lian is a research professor in the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei. He received the B.E. and Ph.D. degrees in computer science from University of Science and Technology of China (USTC) in 2009 and 2014, respectively. His research interest includes spatial data mining, recommender systems, and learning to hash.

Xing Xie (SM'09) is currently a principle researcher in Microsoft Research Asia, and a guest PhD advisor with USTC. His research interest include spatial data mining, location-based services, social networks, and ubiquitous computing. In recent years, he was involved in the program or organizing committees of more than 70 conferences including KDD, WWW, Ubicomp and WSDM.

Enhong Chen (SM'07) received the PhD degree from USTC. He is a professor and vice dean of the School of Computer Science, USTC. His general area of research includes data mining and machine learning, social network analysis, and recommender systems. He was on program committees of numerous conferences including KDD, ICDM, and SDM. His research is supported by the National Science Foundation for Distinguished Young Scholars of China.