Multi-Feature Discrete Collaborative Filtering for Fast Cold-Start Recommendation

Yang Xu,[1] Lei Zhu,[1]* Zhiyong Cheng,[2] Jingjing Li,[3] Jiande Sun[1]
[1]Shandong Normal University
[2]Shandong Computer Science Center (National Supercomputer Center in Jinan)
[2]Qilu University of Technology (Shandong Academy of Sciences)
[3]University of Electronic Science and Technology of China
leizhu0608@gmail.com

Abstract

Hashing is an effective technique to address the large-scale recommendation problem, due to its high computation and storage efficiency on calculating the user preferences on items. However, existing hashing-based recommendation methods still suffer from two important problems: 1) Their recommendation process mainly relies on the user-item interactions and single specific content feature. When the interaction history or the content feature is unavailable (the *cold-start* problem), their performance will be seriously deteriorated. 2) Existing methods learn the hash codes with relaxed optimization or adopt discrete coordinate descent to directly solve binary hash codes, which results in significant quantization loss or consumes considerable computation time. In this paper, we propose a fast cold-start recommendation method, called *Multi-Feature Discrete Collaborative Filtering* (MFDCF), to solve these problems. Specifically, a low-rank self-weighted multi-feature fusion module is designed to adaptively project the multiple content features into binary yet informative hash codes by fully exploiting their complementarity. Additionally, we develop a fast discrete optimization algorithm to directly compute the binary hash codes with simple operations. Experiments on two public recommendation datasets demonstrate that MFDCF outperforms the state-of-the-arts on various aspects.

Introduction

With the development of online applications, recommender systems have been widely adopted by many online services for helping their users find desirable items. However, it is still challenging to accurately and efficiently match items to their potential users, particularly with the ever-growing scales of items and users (Batmaz et al. 2019).

In the past, Collaborative Filtering (CF), as exemplified by Matrix Factorization (MF) algorithms (Koren, Bell, and Volinsky 2009) have demonstrated great successes in both academia and industry. MF factorizes an $n \times m$ user-item rating matrix to project both users and items into a r-dimensional latent feature space, where the user's preference scores for items are predicted by the inner product between their latent features. However, the time complexity for generating top-k items recommendation for all users is $\mathcal{O}(nmr + nm \log k)$ (Zhang, Lian, and Yang 2017). Therefore, MF-based methods are often computational expensive and inefficient when handling the large-scale recommendation applications (Cheng et al. 2018; 2019).

Recent studies show that the hashing-based recommendation algorithms, which encode both users and items into binary codes in Hamming space, are promising to tackle the efficiency challenge (Zhang et al. 2016; 2014). In these methods, the preference score could be efficiently computed by Hamming distance. However, learning binary codes is generally NP-hard (Håstad 2001) due to the discrete constraints. To tackle this problem, the researchers resort to a two-stage hash learning procedure (Liu et al. 2014; Zhang et al. 2014): relaxed optimization and binary quantization. Continuous representations are first computed by the relaxed optimization, and subsequently the hash codes are generated by binary quantization. This learning strategy indeed simplifies the optimization challenge. However, it inevitably suffers from significant quantization loss according to (Zhang et al. 2016). Hence, several solutions are developed to directly optimizing the binary hash codes from the matrix factorization with discrete constraints. Despite much progress has been achieved, they still suffer from two problems: 1) Their recommendation process mainly relies on the user-item interactions and single specific content feature. Under such circumstances, they cannot provide meaningful recommendations for new users (e.g. for the new users who have no interaction history with the items). 2) They learn the hash codes with Discrete Coordinate Descent (DCD) that learns the hash codes bit-by-bit, which results in significant quantization loss or consumes considerable computation time.

In this paper, we propose a fast cold-start recommendation method, called *Multi-Feature Discrete Collaborative Filtering* (MFDCF) to alleviate these problems. Specifically, we propose a low-rank self-weighted multi-feature fusion module to adaptively preserve the multiple content features of users into the compact yet informative hash codes by sufficiently exploiting their complementarity. Our method is inspired by the success of the multiple feature fusion in other relevant areas (Wang et al. 2018; 2012; Lu et al. 2019a;

*The corresponding author

Zhu et al. 2017a). Further, we develop an efficient discrete optimization approach to directly solve binary hash codes by simple efficient operations without quantization errors. Finally, we evaluate the proposed method on two public recommendation datasets, and demonstrate its superior performance over state-of-the-art competing baselines.

The main contributions of this paper are summarized as follows:

- We propose a Multi-Feature Discrete Collaborative Filtering (MFDCF) method to alleviate the cold-start recommendation problem. MFDCF directly and adaptively projects the multiple content features of users into binary hash codes by sufficiently exploiting their complementarity. To the best of our knowledge, there is still no similar work.
- We develop an efficient discrete optimization strategy to directly learn the binary hash codes without relaxed quantization. This strategy avoids performance penalties from both the widely adopted discrete coordinate descent and the storage cost of huge interaction matrix.
- We design a feature-adaptive hash code generation strategy to generate user hash codes that accurately capture the dynamic variations of cold-start user features. Experiments on the public recommendation datasets demonstrate the superior performance of the proposed method over the state-of-the-arts.

Related Work

In this paper, we investigate the hashing-based collaborative filtering at the presence of multiple content features for fast cold-start recommendation. Hence, in this section, we mainly review the recent advanced hashing-based recommendation and cold-start recommendation methods.

A pioneer work, (Das et al. 2007) is proposed to exploit Locality-Sensitive Hashing (LSH) (Gionis, Indyk, and Motwani 1999) to generate hash codes for Google new readers based on their item-sharing history similarity. Based on this, (Karatzoglou, Smola, and Weimer 2010; Zhou and Zha 2012) followed the idea of Iterative Quantization (Gong et al. 2013) to project real latent representations into hash codes. To enhance discriminative capability of hash codes, de-correlation constraint (Liu et al. 2014) and Constant Feature Norm (CFN) constraint (Zhang et al. 2014) are imposed when learning user/item latent representations. The above works basically follow a two-step learning strategy: relaxed optimization and binary quantization. As indicated by (Zhang et al. 2016), this two-step approach will suffer from significant quantization loss.

To alleviate quantization loss, direct binary code learning by discrete optimization is proposed (Shen et al. 2015). In the recommendation area, Discrete Collaborative Filtering (DCF) (Zhang et al. 2016) is the first binarized collaborative filtering method and demonstrates superior performance over aforementioned two-stage recommendation methods. However, it is not applicable to cold-start recommendation scenarios. To address cold-start problem, on the basis of DCF, Discrete Deep Learning (DDL) (Zhang et al. 2018) applies Deep Belief Network (DBN) to extract item representation from item content information, and combines the DBN with DCF. Discrete content-aware matrix factorization methods (Lian et al. 2017; Lian, Xie, and Chen 2019) develop discrete optimization algorithms to learn binary codes for users and items at the presence of their respective content information. Discrete Factorization Machines (DFM) (Liu et al. 2018) learns hash codes for any side feature and models the pair-wise interactions between feature codes. Besides, since the above binary cold-start recommendation frameworks solve the hash codes with bit-by-bit discrete optimization, they still consumes considerable computation time.

The Proposed Method

Notations. Throughout this paper, we utilize bold lowercase letters to represent vectors and bold uppercase letters to represent matrices. All of the vectors in this paper denote column vectors. Non-bold letters represent scalars. We denote $tr(\cdot)$ as the trace of a matrix and $\|\cdot\|_F$ as the Frobenius norm of a matrix. We denote $sgn(\cdot) : R \rightarrow \pm 1$ as the round-off function.

Low-rank Self-weighted Multi-Feature Fusion

Given a training dataset $O = o_i|_{i=1}^n$, which contains n user's multiple features information represented with M different content features (e.g. demographic information such as age, gender, occupation, and interaction preference extracted from item side information). The m-th content feature is $\mathbf{X}^{(m)} = [\mathbf{x}_1^{(m)}, ..., \mathbf{x}_n^{(m)}] \in \mathbb{R}^{d_m \times n}$, where d_m is the dimensionality of the m-th content feature. Since the user's multiple content features are quite diverse and heterogeneous, in this paper, we aim at adaptively mapping multiple content features $\mathbf{X}^{(m)}|_{m=1}^M$ into a consensus multi-feature representation $\mathbf{H} \in \mathbb{R}^{r \times n}$ (r is the hash code length) in a shared homogeneous space. Specifically, it is important to consider the complementarity of multiple content features and the generalization ability of the fusion module. Motivated by these considerations, we introduce a self-weighted fusion strategy and then formulate the multi-feature fusion part as:

$$\min_{\mathbf{W}^{(m)}, \mathbf{H}} \sum_{m=1}^{M} ||\boldsymbol{H} - \boldsymbol{W}^{(m)} \boldsymbol{X}^{(m)}||_F \tag{1}$$

where $||\cdot||_F$ is the Frobenius norm of the matrix. $\boldsymbol{W}^{(m)} \in \mathbb{R}^{r \times d_m}, m = 1, ..., M$ is the mapping matrix of the m-th content feature, $\boldsymbol{H} \in \mathbb{R}^{r \times n}$ is the consensus multi-feature representation. According to (Lu et al. 2019b), Eq.(1) is equivalent to

$$\min_{\boldsymbol{\mu} \in \Delta_M, \mathbf{W}^{(m)}, \mathbf{H}} \sum_{m=1}^{M} \frac{1}{\mu^{(m)}} ||\boldsymbol{H} - \boldsymbol{W}^{(m)} \boldsymbol{X}^{(m)}||_F^2 \tag{2}$$

where $\mu^{(m)}$ is the weight of the m-th content feature and it measures the importance of the current content feature. $\Delta_M \overset{def}{=} \{x \in \mathbb{R}^M | x_i \geq 0, \mathbf{1}^\top x = 1\}$ is the probabilistic simplex.

In real-world recommender systems, such as Taobao[1] and Amazon[2], there are many different kinds of users and items, which have rich and diverse characteristics. However, a specific user only has a small number of interactions in the system with limited items. Consequently, the side information of users and items would be pretty sparse. We need to handle a very high-dimensional and sparse feature matrix. To avoid spurious correlations caused by the mapping matrix, we impose a low-rank constraint on $\boldsymbol{W}$:

$$\min_{\boldsymbol{\mu}\in\Delta_M,\mathbf{W}^{(m)},\mathbf{H}} \sum_{m=1}^{M} \frac{1}{\mu^{(m)}} ||\boldsymbol{H} - \boldsymbol{W}^{(m)}\boldsymbol{X}^{(m)}||_F^2 + \gamma rank(\boldsymbol{W}^{(m)}) \tag{3}$$

where γ is a penalty parameter and $rank(\cdot)$ is the rank operator of a matrix. The low-rank constraint on $\boldsymbol{W}$ helps highlight the latent shared features across different users and handles the extremely spare observations. Meanwhile, the low-rank constraint on $\boldsymbol{W}$ makes the optimization more difficult. To tackle this problem, we adopt an explicit form of low-rank constraint as follows:

$$\begin{aligned}\min_{\boldsymbol{\mu}\in\Delta_M,\mathbf{W}^{(m)},\mathbf{H}} &\sum_{m=1}^{M} \frac{1}{\mu^{(m)}} ||\boldsymbol{H} - \boldsymbol{W}^{(m)}\boldsymbol{X}^{(m)}||_F^2 \\ &+ \gamma \sum_{m=1}^{M}\sum_{i=k+1}^{l} (\sigma_i(\boldsymbol{W^{(m)}}))^2\end{aligned} \tag{4}$$

where l is the total number of singular values of $\boldsymbol{W^{(m)}}$ and $\sigma_i(\boldsymbol{W^{(m)}})$ represents the i-th singular value of $\boldsymbol{W^{(m)}}$. Note that

$$\sum_{i=l+1}^{r} (\sigma_i(\boldsymbol{W^{(m)}}))^2 = tr({\boldsymbol{V}^{(m)}}^\top \boldsymbol{W}^{(m)} {\boldsymbol{W}^{(m)}}^\top \boldsymbol{V}^{(m)}) \tag{5}$$

where $\boldsymbol{V}$ consists of the singular vectors which correspond to the $(r-l)$-smallest singular values of $\boldsymbol{W}^{(m)}{\boldsymbol{W}^{(m)}}^\top$. Thus, the multiple content features fusion module can be rewritten as:

$$\begin{aligned}\min_{\boldsymbol{\mu}\in\Delta_M\mathbf{W}^{(m)},\mathbf{H}} &\sum_{m=1}^{M} \frac{1}{\mu^{(m)}} ||\boldsymbol{H} - \boldsymbol{W}^{(m)}\boldsymbol{X}^{(m)}||_F^2 \\ &+ \gamma \sum_{m=1}^{M} tr({\boldsymbol{V}^{(m)}}^\top \boldsymbol{W}^{(m)} {\boldsymbol{W}^{(m)}}^\top \boldsymbol{V}^{(m)})\end{aligned} \tag{6}$$

Multi-Feature Discrete Collaborative Filtering

In this paper, we fuse multiple content features into binary hash codes with matrix factorization, which has been proved to be accurate and scalable on addressing the collaborative filtering problems. Discrete collaborative filtering generally maps both users and items into a joint low-dimensional Hamming space where the user-item preference is measured by the Hamming similarity between the binary hash codes.

[1]www.taobao.com

[2]www.amazon.com

Given a user-item rating matrix $\boldsymbol{S}$ of size $n \times m$, where n and m are the number of users and items, respectively. Each entry s_{ij} indicates the rating of a user i for an item j. Let $b_i \in \{\pm 1\}^r$ denote the binary hash codes for the i-th user, and $d_j \in \{\pm 1\}^r$ denote the binary hash codes for the j-th item, the rating of user i for item j is approximated by Hamming similarity $(\frac{1}{2} + \frac{1}{2r} b_i^\top d_j)$. Thus, the goal is to learn user binary matrix $\boldsymbol{B} = [\boldsymbol{b_1}, ..., \boldsymbol{b_n}] \in \{\pm 1\}^{r\times n}$ and item binary matrix $\boldsymbol{D} = [\boldsymbol{d_1}, ..., \boldsymbol{d_m}] \in \{\pm 1\}^{r\times m}$, where $r \ll min(n, m)$ is the hash code length. Similar to the problem of conventional collaborative filtering, the basic discrete collaborative filtering can be formulated as:

$$\begin{aligned}&\min_{\boldsymbol{B},\boldsymbol{D}} ||\boldsymbol{S} - \boldsymbol{B}^\top \boldsymbol{D}||_F^2 \\ &s.t. \boldsymbol{B} \in \{\pm 1\}^{r\times n}, \boldsymbol{D} \in \{\pm 1\}^{r\times m}\end{aligned} \tag{7}$$

To address the sparse and cold-start problem, we integrate multiple content features into the above model, by substituting the user binary feature matrix $\boldsymbol{B}$ with the rotated multi-feature representation $\boldsymbol{RH}$($\boldsymbol{R} \in \mathbb{R}^{r\times r}$ is rotation matrix) and keeping their consistency during the optimization process. The formula is given as follows:

$$\begin{aligned}&\min_{\boldsymbol{B},\boldsymbol{D},\boldsymbol{R}} ||\boldsymbol{S} - \boldsymbol{H}^\top \boldsymbol{R}^\top \boldsymbol{D}||_F^2 + \beta ||\boldsymbol{B} - \boldsymbol{RH}||_F^2 \\ &s.t.\ \boldsymbol{B} \in \{\pm 1\}^{r\times n}, \boldsymbol{D} \in \{\pm 1\}^{r\times m}, \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I_r}\end{aligned} \tag{8}$$

This formulation has three advantages: 1) Only one of the decomposed variable is imposed with discrete constraint. As shown in the optimization part, the hash codes can be learned with a simple $sgn(\cdot)$ operation instead of bit-by-bit discrete optimization used by existing discrete recommendation methods. The second regularization term can guarantee the acceptable information loss. 2) The learned hash codes can reflect user's multiple content features via $\boldsymbol{H}$ and involve the latent interactive features in $\boldsymbol{S}$ simultaneously. 3) We extract user's interactive preference from the side information of their rated items as content features. This design not only avoids the approximation of item binary matrix $\boldsymbol{D}$, reduces the complexity of the proposed model, but also effectively captures the content features of items.

Overall Objective Formulation

By integrating the above two parts into a unified learning framework, we derive the overall objective formulation of Multi-Feature Discrete Collaborative Filtering (MFDCF) as:

$$\begin{aligned}&\min_{\mu^{(m)},\boldsymbol{B},\boldsymbol{D},\boldsymbol{H},\boldsymbol{R},\boldsymbol{V},\boldsymbol{W^{(m)}}} \sum_{m=1}^{M} \frac{1}{\mu^{(m)}} ||\boldsymbol{H} - \boldsymbol{W^{(m)}}\boldsymbol{X^{(m)}}||_F^2 \\ &+ \alpha ||\boldsymbol{S} - \boldsymbol{H}^\top \boldsymbol{R}^\top \boldsymbol{D}||_F^2 + \beta ||\boldsymbol{B} - \boldsymbol{RH}||_F^2 \\ &+ \gamma \sum_{m=1}^{M} tr({\boldsymbol{V}^{(m)}}^\top \boldsymbol{W^{(m)}} {\boldsymbol{W^{(m)}}}^\top \boldsymbol{V^{(m)}}) \\ &s.t.\ \boldsymbol{B} \in \{\pm 1\}^{r\times n}, \boldsymbol{D} \in \{\pm 1\}^{r\times m}, \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I_r}\end{aligned} \tag{9}$$

where α, β, γ are balance parameters. The first term projects multiple content features of users into a shared homogeneous space. The second and third terms minimize the information loss during the process of integrating the multiple

content features with the basic discrete CF. The last term is a low-rank constraint for $\boldsymbol{W^{(m)}}$, which can highlight the latent shared features across different users.

Fast Discrete Optimization

Solving hash codes in Eq.(9) is essentially an NP-hard problem due to the discrete constraint on binary feature matrix. Existing discrete recommendation methods always learn the hash codes bit-by-bit with DCD (Shen et al. 2015). Although this strategy alleviates the quantization loss problem caused by conventional two-step relaxing-rounding optimization strategy, it is still time-consuming.

In this paper, with the favorable support of objective formulation, we propose to directly learn the discrete hash codes with fast optimization. Specifically, different from existing discrete recommendation methods (Zhang et al. 2016; 2018; Lian et al. 2017; Liu et al. 2018), we avoid explicitly computing the user-item rating matrix $\boldsymbol{S}$, and achieve linear computation and storage efficiency. We propose an effective optimization algorithm based on augmented Lagrangian multiplier (ALM) (Lin, Chen, and Ma 2010; Murty 2007). In particular, we introduce an auxiliary variable $\boldsymbol{Z_R}$ to separate the constraint on $\boldsymbol{R}$, and transform the objective function Eq.(9) to an equivalent one that can be tackled more easily. Then the Eq.(9) is transformed as:

$$\begin{aligned}\min_{\boldsymbol{\Theta}} \ & \sum_{m=1}^{M}\frac{1}{\mu^{(m)}}||\boldsymbol{H}-\boldsymbol{W^{(m)}}\boldsymbol{X^{(m)}}||_F^2 \\ & +\alpha||\boldsymbol{S}-\boldsymbol{H}^\top\boldsymbol{R}^\top\boldsymbol{D}||_F^2+\beta||\boldsymbol{B}-\boldsymbol{R}\boldsymbol{H}||_F^2 \\ & +\gamma\sum_{m=1}^{M}tr(\boldsymbol{V^{(m)}}^\top\boldsymbol{W^{(m)}}\boldsymbol{W^{(m)}}^\top\boldsymbol{V^{(m)}}) \\ & +\frac{\lambda}{2}||\boldsymbol{R}-\boldsymbol{Z_R}+\frac{\boldsymbol{G_R}}{\boldsymbol{\lambda}}||_F^2 \\ s.t.\ \ & \boldsymbol{B}\in\pm 1^{r\times n},\boldsymbol{D}\in\pm 1^{r\times m},\boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I_r},\boldsymbol{Z_R}^\top\boldsymbol{Z_R}=\boldsymbol{I_r}\end{aligned} \tag{10}$$

where Θ denotes the variables that need to be solved in the objective function, $G_R \in \mathbb{R}^{r\times r}$ measures the difference between the target and auxiliary variable, $\lambda > 0$ is a balance parameter. With this transformation, we follow the alternative optimization process by updating each of $\boldsymbol{\mu^{(m)}}, \boldsymbol{B}, \boldsymbol{D}, \boldsymbol{H}, \boldsymbol{R}, \boldsymbol{V}, \boldsymbol{W^{(m)}}, \boldsymbol{Z_R}$ and $\boldsymbol{G_R}$, given others fixed.

Step 1: learning μ^m. For convenience, we denote $||\boldsymbol{H}-\boldsymbol{W^{(m)}}\boldsymbol{X^{(m)}}||_F$ as $h^{(m)}$. By fixing the other variables, we ignore the term that is irrelevant to $\mu^{(m)}$. The original problem can be rewritten as:

$$\min_{\mu^{(m)}\geq 0,\boldsymbol{1}^\top\boldsymbol{\mu}=1}\sum_{m=1}^{M}\frac{{h^{(m)}}^2}{\mu^{(m)}} \tag{11}$$

With Cauchy-Schwarz inequality, we derive that

$$\sum_{m=1}^{M}\frac{{h^{(m)}}^2}{\mu^{(m)}}\overset{(a)}{=}(\sum_{m=1}^{M}\frac{{h^{(m)}}^2}{\mu^{(m)}})(\sum_{m=1}^{M}\mu^{(m)})\overset{(b)}{\geq}(\sum_{m=1}^{M}h^{(m)})^2$$

where (a) holds since $\boldsymbol{1}^\top\boldsymbol{\mu}=1$ and the equality in (b) holds when $\sqrt{\mu^{(m)}}\propto\frac{h^{(m)}}{\sqrt{\mu^{(m)}}}$. Since $(\sum_{m=1}^{M}h^{(m)})^2 = const$, we can obtain the optimal $\mu^{(m)}$ in Eq.(11) by

$$\mu^{(m)}=\frac{h^{(m)}}{\sum_{m=1}^{M}h^{(m)}} \tag{12}$$

Step 2: learning $\boldsymbol{W}^m$. Removing the terms that are irrelevant to the $\boldsymbol{W}^m$, the optimization formula is rewritten as

$$\begin{aligned}\min_{\boldsymbol{W^{(m)}}}\sum_{m=1}^{M}\frac{1}{\mu^{(m)}}||\boldsymbol{H}-\boldsymbol{W^{(m)}}\boldsymbol{X^{(m)}}||_F^2 \\ +\gamma\sum_{m=1}^{M}tr(\boldsymbol{V^{(m)}}^\top\boldsymbol{W^{(m)}}\boldsymbol{W^{(m)}}^\top\boldsymbol{V^{(m)}})\end{aligned} \tag{13}$$

We calculate the derivative of Eq.(13) with respect to $\boldsymbol{W}$ and set it to zero,

$$\begin{aligned}&\sum_{m=1}^{M}\frac{1}{\mu^{(m)}}||\boldsymbol{H}-\boldsymbol{W^{(m)}}\boldsymbol{X^{(m)}}||_F^2 \\ &\qquad+\gamma\sum_{m=1}^{M}tr(\boldsymbol{V^{(m)}}^\top\boldsymbol{W^{(m)}}\boldsymbol{W^{(m)}}^\top\boldsymbol{V^{(m)}})=0 \\ \Rightarrow\ &\gamma\boldsymbol{V^{(m)}}\boldsymbol{V^{(m)}}^\top\boldsymbol{W^{(m)}}+\frac{\boldsymbol{1}}{\boldsymbol{\mu^{(m)}}}\boldsymbol{W^{(m)}}\boldsymbol{X^{(m)}}\boldsymbol{X^{(m)}}^\top \\ &\qquad=\frac{\boldsymbol{1}}{\boldsymbol{\mu^{(m)}}}\boldsymbol{H}\boldsymbol{X^{(m)}}^\top\end{aligned} \tag{14}$$

By using the following substitutions,

$$\begin{cases}\boldsymbol{A}=\boldsymbol{\gamma}\boldsymbol{V^{(m)}}\boldsymbol{V^{(m)}}^\top \\ \boldsymbol{B}=\frac{\boldsymbol{1}}{\boldsymbol{\mu^{(m)}}}\boldsymbol{X^{(m)}}\boldsymbol{X^{(m)}}^\top \\ \boldsymbol{C}=\frac{\boldsymbol{1}}{\boldsymbol{\mu^{(m)}}}\boldsymbol{H}\boldsymbol{X^{(m)}}^\top\end{cases} \tag{15}$$

Eq.(14) can be rewritten as $\boldsymbol{AW}+\boldsymbol{WB}=\boldsymbol{C}$, which can be efficiently solved by Sylvester operation in Matlab.

Step 3: learning $\boldsymbol{R}$. Similarly, the optimization formula for updating R can be represented as

$$\begin{aligned}\min_{\boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I_r}}\ & tr(-2\alpha\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{S}^\top\boldsymbol{H}^\top+\alpha\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{D}^\top\boldsymbol{R}\boldsymbol{H}\boldsymbol{H}^\top \\ & -2\beta\boldsymbol{R}^\top\boldsymbol{B}\boldsymbol{H}^\top-\lambda\boldsymbol{R}^\top(\boldsymbol{Z_R}-\frac{\boldsymbol{G_R}}{\lambda}))\end{aligned} \tag{16}$$

We introduce an auxiliary variable $\boldsymbol{Z_R}$ and substitute $\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{D}^\top\boldsymbol{R}\boldsymbol{H}\boldsymbol{H}^\top$ with $\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{D}^\top\boldsymbol{Z_R}\boldsymbol{H}\boldsymbol{H}^\top$, the Eq.(16) can be transformed into the following form

$$\begin{aligned}&\max_{\boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I_r}}\ tr(\boldsymbol{R}^\top\boldsymbol{C}), \\ &\boldsymbol{C}=2\alpha\boldsymbol{D}\boldsymbol{S}^\top\boldsymbol{H}^\top-\alpha\boldsymbol{D}\boldsymbol{D}^\top\boldsymbol{Z_R}\boldsymbol{H}\boldsymbol{H}^\top \\ &+2\beta\boldsymbol{B}\boldsymbol{H}^\top+\lambda\boldsymbol{Z_R}-\boldsymbol{G_R}\end{aligned} \tag{17}$$

The optimal $\boldsymbol{R}$ is defined as $\boldsymbol{R} = \boldsymbol{P}\boldsymbol{Q}^\top$, where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are comprised of left-singular and right-singular vectors of $\boldsymbol{C}$ respectively (Zhu et al. 2017b).

Note that, the user-item rating matrix $\boldsymbol{S} \in \mathbb{R}^{n\times m}$ is included in the term $\boldsymbol{D}\boldsymbol{S}^\top\boldsymbol{H}^\top$ when updating $\boldsymbol{R}$. In real-world retail giants, such as Taobao and Amazon, there are hundreds of millions of users and even more items. In consequence, the user-item rating matrix $\boldsymbol{S}$ would be pretty enormous and sparse. If we compute $\boldsymbol{S}$ directly, the computational complexity will be $\mathcal{O}(mn)$ and it is extremely expensive to calculate and store $\boldsymbol{S}$. In this paper, we apply the singular value decomposition to obtain the left singular and right singular vectors as well as the corresponding singular values of $\boldsymbol{S}$. We utilize a diagonal matrix Σ_o^S to store the o-largest ($o \ll min\{m,n\}$) singular values, and employ an $n \times o$ matrix $\boldsymbol{P}_{\boldsymbol{o}}^{\boldsymbol{S}}$, an $o \times m$ matrix $\boldsymbol{Q}_{\boldsymbol{o}}^{\boldsymbol{S}}$ to store the corresponding left singular and right singular vectors respectively. We substitute $\boldsymbol{S}$ with $\boldsymbol{P}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{\Sigma}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{Q}_{\boldsymbol{o}}^{\boldsymbol{S}}$ and the computational complexity can be reduced to $\mathcal{O}(max\{m,n\})$.

Thus, the calculation of $\boldsymbol{C}$ can be transformed as

$$\begin{aligned}\boldsymbol{C} =& 2\alpha\boldsymbol{D}\boldsymbol{Q}_{\boldsymbol{o}}^{\boldsymbol{S}\top}\boldsymbol{\Sigma}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{P}_{\boldsymbol{o}}^{\boldsymbol{S}\top}\boldsymbol{H}^\top - \alpha\boldsymbol{D}\boldsymbol{D}^\top\boldsymbol{Z}_{\boldsymbol{R}}\boldsymbol{H}\boldsymbol{H}^\top \\ &+ 2\beta\boldsymbol{B}\boldsymbol{H}^\top + \lambda\boldsymbol{Z}_{\boldsymbol{R}} - \boldsymbol{G}_{\boldsymbol{R}}\end{aligned} \tag{18}$$

With Eq.(18), both the computation and storage cost can be decreased with the guarantee of accuracy.

Step 4: learning $\boldsymbol{H}$. We calculate the derivative of objective function with respect to $\boldsymbol{H}$ and set it to zero, then we get

$$\begin{aligned}H =& (\sum_{m=1}^{M}\frac{1}{\mu_{(m)}}\boldsymbol{I}_{\boldsymbol{r}} + \alpha\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{D}^\top\boldsymbol{R} + \beta\boldsymbol{I}_{\boldsymbol{r}})^{-1} \\ & (\sum_{m=1}^{M}\frac{1}{\mu_{(m)}}\boldsymbol{W}^{(\boldsymbol{m})}\boldsymbol{X}^{(\boldsymbol{m})} + \alpha\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{S}^\top + \beta\boldsymbol{R}^\top\boldsymbol{B})\end{aligned} \tag{19}$$

where $\boldsymbol{S}$ is substituted with $\boldsymbol{P}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{\Sigma}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{Q}_{\boldsymbol{o}}^{\boldsymbol{S}}$, and then we have

$$\alpha\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{S}^\top = \alpha\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{Q}_{\boldsymbol{o}}^{\boldsymbol{S}\top}\boldsymbol{\Sigma}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{P}_{\boldsymbol{o}}^{\boldsymbol{S}\top} \tag{20}$$

The time complexity of computing $\boldsymbol{R}^\top\boldsymbol{D}\boldsymbol{S}^\top$ is reduced to $\mathcal{O}(max\{m,n\})$.

Step 5: learning $\boldsymbol{B}, \boldsymbol{D}$. We calculate the derivative of objective function with respect to $\boldsymbol{B}$ and $\boldsymbol{D}$ respectively, and set them to zero, then we can obtain the closed solutions of $\boldsymbol{B}, \boldsymbol{D}$ as

$$\boldsymbol{D} = sgn((\boldsymbol{H}^\top\boldsymbol{R}^\top)^{-1}\boldsymbol{S}), \quad \boldsymbol{B} = sgn(\boldsymbol{R}\boldsymbol{H}) \tag{21}$$

where $\boldsymbol{S}$ is also substituted with $\boldsymbol{P}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{\Sigma}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{Q}_{\boldsymbol{o}}^{\boldsymbol{S}}$, and update rule of $\boldsymbol{D}$ is transformed as

$$\boldsymbol{D} = sgn((\boldsymbol{H}^\top\boldsymbol{R}^\top)^{-1}\boldsymbol{P}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{\Sigma}_{\boldsymbol{o}}^{\boldsymbol{S}}\boldsymbol{Q}_{\boldsymbol{o}}^{\boldsymbol{S}}) \tag{22}$$

Step 6: learning $\boldsymbol{V}^{(\boldsymbol{m})}$. As described in Eq.(5), $\boldsymbol{V}$ is stacked by the singular vectors which correspond to the $(r-l)$-smallest singular values of $\boldsymbol{W}^{(m)}\boldsymbol{W}^{(m)^\top}$. Thus we can solve the eigen-decomposition problem to get $\boldsymbol{V}$:

$$\boldsymbol{V} \leftarrow svd(\boldsymbol{W}^{(m)}\boldsymbol{W}^{(m)^\top}) \tag{23}$$

Step 7: learning $\boldsymbol{Z}_{\boldsymbol{R}}$. The objective function with respect to $\boldsymbol{Z}_{\boldsymbol{R}}$ can be represented as

$$\max_{\boldsymbol{Z}_{\boldsymbol{R}}^\top\boldsymbol{Z}_{\boldsymbol{R}}=\boldsymbol{I}_{\boldsymbol{r}}} tr(\boldsymbol{Z}_{\boldsymbol{R}}^\top\boldsymbol{C}_{\boldsymbol{z}\boldsymbol{r}}) \tag{24}$$

where $\boldsymbol{C}_{\boldsymbol{z}\boldsymbol{r}} = -\alpha\boldsymbol{D}\boldsymbol{D}^\top\boldsymbol{R}\boldsymbol{H}\boldsymbol{H}^\top + \lambda\boldsymbol{R} + \boldsymbol{G}_{\boldsymbol{R}}$. The optimal $\boldsymbol{Z}_{\boldsymbol{R}}$ is defined as $\boldsymbol{Z}_{\boldsymbol{R}} = \boldsymbol{P}_{\boldsymbol{z}\boldsymbol{r}}\boldsymbol{Q}_{\boldsymbol{z}\boldsymbol{r}}^\top$, where $\boldsymbol{P}_{\boldsymbol{z}\boldsymbol{r}}$ and $\boldsymbol{Q}_{\boldsymbol{z}\boldsymbol{r}}^\top$ are comprised of left-singular and right-singular vectors of $\boldsymbol{C}_{\boldsymbol{z}\boldsymbol{r}}$ respectively.

Step 8: learning $\boldsymbol{G}_{\boldsymbol{R}}$. By fixing other variables, the update rule of $\boldsymbol{G}_{\boldsymbol{R}}$ is

$$\boldsymbol{G}_{\boldsymbol{R}} = \boldsymbol{G}_{\boldsymbol{R}} + \lambda(\boldsymbol{R} - \boldsymbol{Z}_{\boldsymbol{R}}) \tag{25}$$

Feature-adaptive Hash Code Generation for Cold-start Users

In the process of online recommendation, we aim to map multiple content features of the target users into binary hash codes with the learned hash projection matrix $\{\boldsymbol{W}^{(\boldsymbol{m})}\}_{m=1}^{M}$. When cold-start users have no rating history in the training set and are only associated with initial demographic information, the fixed feature weights obtained from offline hash code learning cannot address the feature-missing problem.

In this paper, with the support of offline hash learning, we propose to generate hash codes for cold-start users with a self-weighting scheme. The objective function is formulated as

$$\min_{\boldsymbol{B}_{\boldsymbol{u}}\in\{\pm1\}^{r\times n_u}} \sum_{m=1}^{M} ||\boldsymbol{B}_u - \boldsymbol{W}^{(m)}\boldsymbol{X}_u^{(m)}||_F \tag{26}$$

where $\boldsymbol{W}^{(m)} \in \mathbb{R}^{r\times d}$ is the linear projection matrix from Eq.(9), $\boldsymbol{X}_u^{(m)}$ is content feature of target users, and n_u is the number of target users. As proved by (Lu et al. 2019b), Eq.(26) can be shown to be equivalent to

$$\min_{\boldsymbol{B}_{\boldsymbol{u}}\in\{\pm1\}^{r\times n_u}, \boldsymbol{\mu}\in\Delta_M} \sum_{m=1}^{M} \frac{1}{\mu_u^{(m)}} ||\boldsymbol{B}_u - \boldsymbol{W}^{(m)}\boldsymbol{X}_u^{(m)}||_F^2 \tag{27}$$

We employ alternating optimization to update $\mu_u^{(m)}$ and $\boldsymbol{B}_{\boldsymbol{u}}$. The update rules are

$$\begin{aligned}&\mu_u^{(m)} = \frac{h_u^{(m)}}{\sum_{m=1}^{M} h_u^{(m)}}, \; h_u^{(m)} = ||\boldsymbol{B}_u - \boldsymbol{W}^{(m)}\boldsymbol{X}_u^{(m)}||_F \\ &\boldsymbol{B}_{\boldsymbol{u}} = sgn(\sum_{m=1}^{M}\frac{1}{\mu_u^{(m)}}\boldsymbol{W}^{(m)}\boldsymbol{X}_u^{(m)})\end{aligned} \tag{28}$$

Experiments

Evaluation Datasets

We evaluate the proposed method on two public recommendation datasets: Movielens-1M[3] and BookCrossing[4]. In these two datasets, each user has only one rating for an item.

[3] https://grouplens.org/datasets/movielens/

[4] https://grouplens.org/datasets/book-crossing/

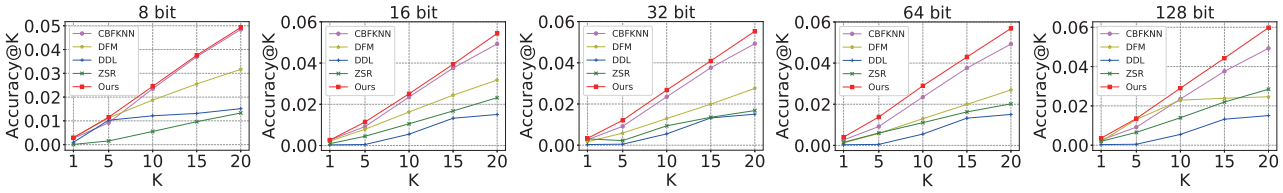

Figure 1: Cold-start recommendation performance on MovieLens-1M.

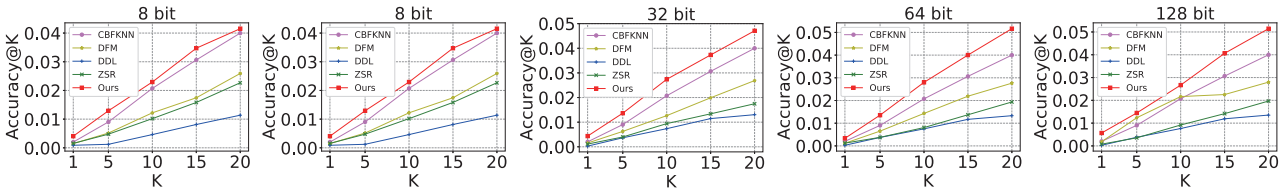

Figure 2: Cold-start recommendation performance on BookCrossing.

- **Movielens-1M**: This dataset is collected from the MovieLens website by GroupLens Research. It originally includes 1,000,000 ratings from 6040 users for 3952 movies. The rating score is from 1 to 5 with 1 granularity. The users in this dataset are associated with demographic information (e.g. gender, age, and occupation), and the movies are related to 3-5 labels from a dictionary of 18 genre labels.
- **BookCrossing**: This dataset is collected by Cai-Nicolas Ziegler from the Book-Crossing community. It contains 278,858 users providing 1,149,780 ratings (contain implicit and explicit feedback) about 271,379 books. The rating score is from 1 to 10 with 1 interval for explicit feedback, or expressed by 0 for implicit feedback. Most users in this dataset are associated with demographic information (e.g. age and location).

Considering the extreme sparsity of the original BookCrossing dataset, we remove the users with less than 20 ratings and the items rated by less than 20 users. After the filtering, there are 2,151 users, 6,830 items, and 180,595 ratings left in the BookCrossing dataset. For the MovieLens-1M dataset, we keep all users and items without any filtering. The statistics of the datasets are summarized in Table 2. The bag-of-words encoding method is used to extract the side information of the item, and one-hot encoding approach is adopted to generate feature representation of user's demographic information. To accelerate the running speed, we follow (Wang et al. 2017) and perform PCA to reduce the interactive preference feature dimension to 128. In our experiments, we randomly select 20% users as cold-start users, and their ratings are removed. We repeat the experiments with 5 random splits and report the average values as the experimental results.

Evaluation Metrics

The goal of our proposed method is to find out the top-k items that user may be interested in. In our experiment, we adopt the evaluation metric Accuracy@k (Zhang et al. 2018; Du et al. 2018) to evaluate whether the target user's favorite items appear in the top-k recommendation list.

Table 1: Statistics of experimental datasets.

Dataset	#User	#Item	#Rating	Sparsity
MovieLens-1M	6,040	3,952	1,000,209	95.81%
BookCrossing	2,151	6,830	180,595	98.77%

Accuracy@k is to test whether the target user's favorite items that appears in the top-k recommendation list. Given the value of k, similar to (Zhang et al. 2018; Du et al. 2018), we calculate Accuracy@k value as:

$$Accuracy@k = \frac{\#Hit@k}{|D_{test}|} \tag{29}$$

where $|D_{test}|$ is the number of test cases, and $\#Hit@k$ is the total number of hits in the test set.

Evaluation Baselines

In this paper, we compare our approach with two state-of-the-art continuous value based recommendation methods and two hashing based binary recommendation methods.

- **Content Based Filtering KNN (CBFKNN)** (Gantner et al. 2010) is a straightforward cold-start recommendation approach based on the user similarity. Specifically, it adopts the low-dimensional projection of user attribute to calculate the user simialrity.
- **Discrete Factorization Machines (DFM)** (Liu et al. 2018) is the first binarized factorization machines method that learns the hash codes for any side feature and models the pair-wise interaction between feature codes.
- **Discrete Deep Learning (DDL)** (Zhang et al. 2018) is a binary deep recommendation approach. It adopts Deep Belief Network to extract item representation from item side information, and combines the DBN with DCF to solve the cold-start recommendation problem.

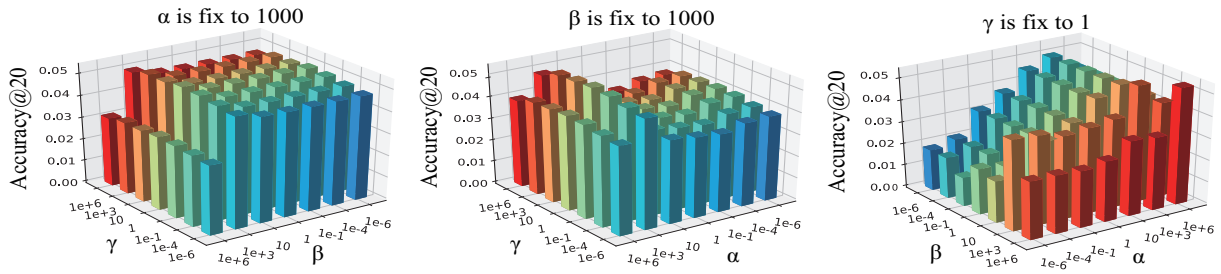

Figure 3: Performance variations with the key parameters on MovieLens-1M

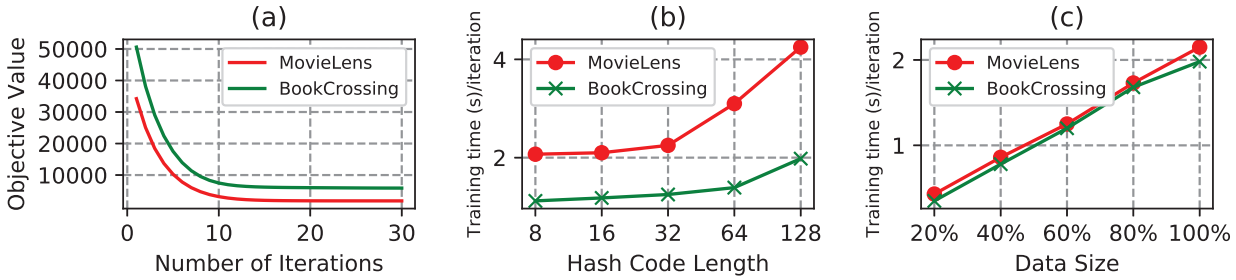

Figure 4: (a) Convergence curve, (b-c) Efficiency v.s hash code length and data size.

Table 2: Training time comparison with two hashing-based recommendation approaches on MovieLens.

Method/#Bits	8	16	32	64	128
DDL	3148.86	3206.09	3289.81	3372.19	3855.81
DFM	166.55	170.86	172.75	196.45	246.5
Ours	**58.87**	**60.49**	**62.55**	**68.41**	**90.17**

- **Zero-Shot Recommendation (ZSR)** (Li et al. 2019b) considers cold-start recommendation problem as a zero-shot learning problem (Li et al. 2019a). It extracts user preference for each item from user attribute.

In experiments, we adopt 5-fold cross validation method on random split of training data to tune the optimal hyper-parameters of all compared approaches. All the best hyper-parameters are found by grid search.

Accuracy Comparison

In this subseciton, we evaluate the recommendation accuracy of MFDCF and the baselines in cold-start recommendation scenario. Figure 1 and 2 demonstrate the Accuracy@k of the compared approaches on two real-world recommendation datasets for the cold-start recommendation task. Compared with existing hashing-based recommendation approaches, the proposed MFDCF consistently outperforms the compared baselines. DFM exploits the factorization machine to model the potential relevance between user characteristics and product features. However, it ignores the collaborative interaction. DDL is based on the discrete collaborative filtering. It adopts DBN to generate item feature representation from their side information. Nevertheless, the structure of DBN is independent with the overall optimization process, which limits the learning capability of DDL. Additionally, these experimental results show that the proposed MFDCF outperforms the compared continuous value based hybrid recommendation methods under the same cold-start settings. The better performance of MFDCF than CBFKNN and ZSR validates the effects of the proposed multiple feature fusion strategy.

Model Analysis

Parameter and convergence sensitivity analysis. We conduct experiments to observe the performance variations with the involved parameters α, β, γ. We fix the hash code length as 128 bits and report results on MovieLens-1M. Similar results can be found on other datasets and hash code lengths. Since α, β, and γ are equipped in the same objective function, we change their values from the range of $\{10^{-6}, 10^{-4}, 10^{-1}, 1, 10, 10^{3}, 10^{6}\}$ while fixing other parameters. Detailed experimental results are presented in Figure 5. From it, we can observe that the performance is relatively better when α is in the range of $\{10^{6}, 10^{3}, 10\}$, β is in the range of $\{10^{3}, 10\}$, and γ is in the range of $\{1, 10\}$. The performance variations with γ shows that the low-rank constraint is well on highlighting the latent shared features across different users. The convergence curves recording the objective function of MFDCF method with the number of iterations are shown in Figure 4(a). This experiment result indicates that our proposed method converges very fast.

Efficiency v.s. hash code length and data size. We conduct the experiments to investigate the efficiency variations of MFDCF with the increase of hash code length and training data size on two datasets. The average time cost of training iteration is shown in Figure 4(b-c). When the hash code

length is fixed as 32, each round of training iteration costs several seconds and scales linearly with the increase of data size. When running MFDCF on 100% training data, each round of iteration scales quadratically with the increase of code length due to the time complexity of optimization process is $\mathcal{O}(max\{mr^2, nr^2\})$.

Run time comparison. In this experiment, we compare the computation efficiency of our approach with two state-of-the-art hashing-based recommendation methods DMF and DDL. Table 2 demonstrates the training time of these methods on MovieLens-1M using a 3.4GHz Intel® Core(TM) i7-6700 CPU. Compared with DDL and DFM, our MFDCF is about 50 and 3 times faster respectively. The superior performance of the proposed method is attributed to that both DDL and DFM iteratively learn the hash codes bit-by-bit with discrete coordinate descent. Additionally, DDL requires to update the parameters of DBN iteratively, which consumes more time.

Conclusion

In this paper, we design a unified multi-feature discrete collaborative filtering method that projects multiple content features of users into the binary hash codes to support fast cold-start recommendation. Our model has four advantages: 1) handles the data sparsity problem with low-rank constraint. 2) enhances the discriminative capability of hash codes with multi-feature binary embedding. 3) generates feature-adaptive hash codes for varied cold-start users. 4) achieves computation and storage efficient discrete binary optimization. Experiments on two public recommendation datasets demonstrate the state-of-the-art performance of the proposed method.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive and helpful suggestions. The work is partially supported by the National Natural Science Foundation of China (61802236, 61902223, U1836216, U1736122), in part by the Natural Science Foundation of Shandong, China (No. ZR2019QF002), in part by the Youth Innovation Project of Shandong Universities, China (No. 2019KJN040), in part by the Natural Science Foundation for Distinguished Young Scholars of Shandong Province (JQ201718), and in part by Taishan Scholar Project of Shandong, China.

References

Batmaz, Z.; Yurekli, A.; Bilge, A.; and Kaleli, C. 2019. A review on deep learning for recommender systems: challenges and remedies. *Artif. Intell. Rev.* 52(1):1–37.

Cheng, Z.; Ding, Y.; He, X.; Zhu, L.; Song, X.; and Kankanhalli, M. S. 2018. Aˆ3ncf: An adaptive aspect attention model for rating prediction. In *IJCAI*, 3748–3754.

Cheng, Z.; Chang, X.; Zhu, L.; Catherine Kanjirathinkal, R.; and Kankanhalli, M. S. 2019. MMALFM: explainable recommendation by leveraging reviews and images. *TOIS* 37(2):1–28.

Das, A.; Datar, M.; Garg, A.; and Rajaram, S. 2007. Google news personalization: scalable online collaborative filtering. In *WWW*, 271–280.

Du, X.; Yin, H.; Chen, L.; Wang, Y.; and Zhou, X. 2018. Personalized video recommendation using rich contents from videos. *TKDE*.

Gantner, Z.; Drumond, L.; Freudenthaler, C.; Rendle, S.; and Schmidt-Thieme, L. 2010. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*, 176–185.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity search in high dimensions via hashing. In *VLDB*, 518–529.

Gong, Y.; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI* 35(12):2916–2929.

Håstad, J. 2001. Some optimal inapproximability results. *J. ACM* 48(4):798–859.

Karatzoglou, A.; Smola, A. J.; and Weimer, M. 2010. Collaborative filtering on a budget. In *AISTATS*, 389–396.

Koren, Y.; Bell, R. M.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* 42(8):30–37.

Li, J.; Jing, M.; Lu, K.; Ding, Z.; Zhu, L.; and Huang, Z. 2019a. Leveraging the invariant side of generative zero-shot learning. In *CVPR*, 7402–7411.

Li, J.; Jing, M.; Lu, K.; Zhu, L.; Yang, Y.; and Huang, Z. 2019b. From zero-shot learning to cold-start recommendation. In *AAAI*, 4189–4196.

Lian, D.; Liu, R.; Ge, Y.; Zheng, K.; Xie, X.; and Cao, L. 2017. Discrete content-aware matrix factorization. In *KDD*, 325–334.

Lian, D.; Xie, X.; and Chen, E. 2019. Discrete matrix factorization and extension for fast item recommendation. *TKDE* DOI: 10.1109/TKDE.2019.2951386.

Lin, Z.; Chen, M.; and Ma, Y. 2010. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *CoRR* abs/1009.5055.

Liu, X.; He, J.; Deng, C.; and Lang, B. 2014. Collaborative hashing. In *CVPR*, 2147–2154.

Liu, H.; He, X.; Feng, F.; Nie, L.; Liu, R.; and Zhang, H. 2018. Discrete factorization machines for fast feature-based recommendation. In *IJCAI*, 3449–3455.

Lu, X.; Zhu, L.; Cheng, Z.; Li, J.; Nie, X.; and Zhang, H. 2019a. Flexible online multi-modal hashing for large-scale multimedia retrieval. In *MM*, 1129–1137.

Lu, X.; Zhu, L.; Cheng, Z.; Nie, L.; and Zhang, H. 2019b. Online multi-modal hashing with dynamic query-adaption. In *SIGIR*, 715–724.

Murty, K. G. 2007. Nonlinear programming theory and algorithms. *Technometrics* 49(1):105.

Shen, F.; Shen, C.; Liu, W.; and Shen, H. T. 2015. Supervised discrete hashing. In *CVPR*, 37–45.

Wang, M.; Li, H.; Tao, D.; Lu, K.; and Wu, X. 2012. Multimodal graph-based reranking for web image search. *TIP* 21(11):4649–4661.

Wang, M.; Fu, W.; Hao, S.; Liu, H.; and Wu, X. 2017. Learning on big graph: Label inference and regularization with anchor hierarchy. *TKDE* 29(5):1101–1114.

Wang, M.; Luo, C.; Ni, B.; Yuan, J.; Wang, J.; and Yan, S. 2018. First-person daily activity recognition with manipulated object proposals and non-linear feature fusion. *TCSVT* 28(10):2946–2955.

Zhang, Z.; Wang, Q.; Ruan, L.; and Si, L. 2014. Preference preserving hashing for efficient recommendation. In *SIGIR*, 183–192.

Zhang, H.; Shen, F.; Liu, W.; He, X.; Luan, H.; and Chua, T. 2016. Discrete collaborative filtering. In *SIGIR*, 325–334.

Zhang, Y.; Yin, H.; Huang, Z.; Du, X.; Yang, G.; and Lian, D. 2018. Discrete deep learning for fast content-aware recommendation. In *WSDM*, 717–726.

Zhang, Y.; Lian, D.; and Yang, G. 2017. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *AAAI*, 1669–1675.

Zhou, K., and Zha, H. 2012. Learning binary codes for collaborative filtering. In *KDD*, 498–506.

Zhu, L.; Huang, Z.; Liu, X.; He, X.; Song, J.; and Zhou, X. 2017a. Discrete multi-modal hashing with canonical views for robust mobile landmark search. *TMM* 19(9):2066–2079.

Zhu, L.; Shen, J.; Xie, L.; and Cheng, Z. 2017b. Unsupervised visual hashing with semantic assistant for content-based image retrieval. *TKDE* 29(2):472–486.