Binarized Collaborative Filtering with Distilling Graph Convolutional Networks

Haoyu Wang[1,2] , **Defu Lian**[1*] and **Yong Ge**[3]

[1]School of Computer Science and Technology, University of Science and Technology of China
[2]University of Electronic Science and Technology of China
[3]University of Arizona
{dove.ustc, haoyu.uestc}@gmail.com, yongge@email.arizona.edu

Abstract

The efficiency of top-K item recommendation based on implicit feedback are vital to recommender systems in real world, but it is very challenging due to the lack of negative samples and the large number of candidate items. To address the challenges, we firstly introduce an improved Graph Convolutional Network (GCN) model with high-order feature interaction considered. Then we distill the ranking information derived from GCN into binarized collaborative filtering, which makes use of binary representation to improve the efficiency of online recommendation. However, binary codes are not only hard to be optimized but also likely to incur the loss of information during the training processing. Therefore, we propose a novel framework to convert the binary constrained optimization problem into an equivalent continuous optimization problem with a stochastic penalty. The binarized collaborative filtering model is then easily optimized by many popular solvers like SGD and Adam. The proposed algorithm is finally evaluated on three real-world datasets and shown the superiority to the competing baselines.

1 Introduction

Nowadays, recommender systerms are widely used in people's daily life [Liu *et al.*, 2011; Lian *et al.*, 2016; Li *et al.*, 2018a], but a growing scale of users and products renders recommendation challenging. Because implicit feedback is more common and easier to collect than explicit feedback, we concentrate on how to accelerate top-K recommendation based on implicit feedback. However, there are two challengings to address. Firstly, compared with explicit feedback, implicit feedback is more difficult to utilize because of the lack of negative samples [Pan *et al.*, 2008]. Secondly, generating top-k preferred items for each user is extremely time-consuming.

For the first problem, recently, SpectralCF [Zheng *et al.*, 2018] combined collaborative filtering model with graph convolutional network [Henaff *et al.*, 2015] to mine hidden interactions between users and items from spectral domain, which showed enormous potential for implicit feedback problem [Zheng *et al.*, 2018]. However, SpectralCF ignores high-order feature interaction.

For the second problem, for extracting top-K preferred items for each user, the time complexity of recommendation is $\mathcal{O}(MND + MN\log K)$ when there are M users, N items and D dimensions in the latent space. Therefore, this is a critical efficiency bottleneck. However, it is necessary to timely update recommendation algorithms and the recommendation list because user interest evolves frequently. Fortunately, hash technique, encoding real-valued vectors/matrices into binary codes($e.g.,\{0,1\},\{-1,1\}$), is promising to address this challenge because inner product can be efficiently computed between binary codes via bit operation. Finding approximate top-K items can be even finished in sublinear or logarithmic time [Wang *et al.*, 2012; Muja and Lowe, 2009] by making use of index technique.

Several methods applied hash techniques to recommendation. Some two-stage approximation methods like BCCF [Zhou and Zha, 2012], PPH [Zhang *et al.*, 2014], CH [Liu *et al.*, 2014] incur large quantization loss [Zhang *et al.*, 2016], and a direct optimization model DCF [Zhang *et al.*, 2016] is easy to fall into a local optimum because it is based on local search. To this end, to improve the accuracy of hashing-based recommender systems for implicit feedback, we propose a binarized collaborative filtering framework with distilling graph convolutional network. In the framework, we firstly train a CF-based GCN model (GCN-CF) which can capture high-order feature interaction via cross operation. Following that, we distill the ranking information from the trained GCN-CF model into a binarized model (DGCN-BinCF) with knowledge distillation technique (KD [Hinton *et al.*, 2015]). To be more specific, we introduce a novel distillation loss, which penalizes not only the discrepancy between distributions of positive items defined by GCN-CF and that defined by Bin-CF, but also the the discrepancy between distributions of sampled negative items. Noting that learning hash codes is generally NP-hard [Håstad, 2001], approximation methods are appropriate choices but it may incur the loss of information during the training process. To this end, inspired by [Dai *et al.*, 2016], we transform the binary optimization problem to an equivalent continuous optimization problem by imposing a stochastic penalty term. Therefore,

*Corresponding author

any gradient-based solver can optimize the overall loss with ranking-based loss with knowledge distillation loss.

Our contributions are summarized as follows:

- We propose a novel framework DGCN-BinCF to distill the ranking information from the proposed GCN-CF model into the binary model. To the best of our knowledge, DGCN-BinCF is the first model utilizing knowledge distilling to improve the performance of binarized model. We also improve GCN via adding a cross operation to aggregate users and items' own high-order features.
- We propose a generic method to relax the binary constraint problem to an equivalent bound-constrained continuous optimization problem. Hence, we can optimize the original problem by popular solvers directly.
- Through extensive experiments performed on three real-world datasets, we show the superiority of the proposed framework to the state-of-the-art baselines.

2 Related Work

In this section, we review several works related to our task including GCN for recommender systems, recent hashing-based collaborative filtering methods and distilling knowledge techniques for ranking.

2.1 GCN for Recommender Systems

How to take advantage of the rich linkage information from the user-item bipartite graph is crucial for implicit feedback. Some work used GCN to solve it such as SpectralCF [Zheng *et al.*, 2018], GCMC [Berg *et al.*, 2017], RMGCNN [Monti *et al.*, 2017], GCNWSRS [Ying *et al.*, 2018], LGCN [Gao *et al.*, 2018], etc. (1)SpectralCF was the first model to learn from the Spectral domain of the user-item bipartite graph directly based on collaborative filtering. Because it could discover deep connections between users and items, it may alleviate cold-start problem. (2)GCMC combined GCN model with a graph auto-encoder to learn users' and items' latent factors. (3)RMGCNN proposed a matrix completion architecture combining a multi-graph convolutional neural network with a recurrent neural network. (4)GCNWSRS focused on how to apply GCN model for web-scale recommendation tasks effectively, like billion of items and hundreds of millions of users. (5)LGCN proposed a sub-graph training strategy to save memory and computational resource requirements greatly. Its experiments showed it was more efficient as compared to prior approaches.

2.2 Discrete Hashing for Collaborative Filtering

A pioneer work was to exploit Locality-Sensitive Hashing [Datar *et al.*, 2004] to generate binary codes for Google News readers according to their click history [Das *et al.*, 2007]. Then [Karatzoglou *et al.*, 2010] proposed a method mapping users and items' latent factors into Hamming space to obtain binary representation. Later, following this, some two stage methods [Zhou and Zha, 2012; Zhang *et al.*, 2014] which relaxed binary constraints at first and then quantified binary codes. Nonetheless, [Zhang *et al.*, 2016] proposed that those two-stage methods suffered from large quantization loss. Therefore, DCF proposed a method which could optimize binary codes directly. However, DCF optimizes binary codes via searching neighborhoods with the distance one. So it is easy to fall into local optima.

2.3 Distilling Knowledge for Ranking

[Hinton *et al.*, 2015] was the first one that proposed method "Knowledge Distilling", which trained a complex neural network firstly and then transferred the complex model to a small model. The role of the complex model is similar to a teacher, and the role of the small model is similar to a student. Following this, DarkRank [Chen *et al.*, 2018] proposed a method combining deep metric learning and "Learning to rank" technique with KD to solve pedestrian re-identification, image retrieval and image clustering tasks. In addition, [Tang and Wang, 2018] applied KD with point-wise ranking on recommendation task. Unfortunately, it did not focus on implicit feedback problem and how to transfer unobserved interaction information.

3 Definitions and Preliminaries

Throughout the paper, we denote vectors by boldfaced lowercase letters and matrices by boldfaced uppercase letters. All vectors are considered as column vectors. Next, we define the following definitions in this paper:

Definition 1 *(Bipartite Graph)A user-item bipartite graph with $M + N$ vertices and E edges is defined as $\mathcal{G} = \{\mathcal{U}, \mathcal{I}, \mathcal{E}\}$, where $\mathcal{U}$ and $\mathcal{I}$ are two disjoint vertex sets of user and item, and $M = |\mathcal{U}|$, $N = |\mathcal{I}|$. For each edge $e \in \mathcal{E}$, it has the form that $e = (u, i)$, where $u \in \mathcal{U}$ and $i \in \mathcal{I}$, which shows that there exists an interaction between user u and item i in the training set.*

Definition 2 *(Laplacian Matrix)Given a bipartite graph with $M+N$ vertices and E edges, the laplacian matrix $\boldsymbol{L}$ is defined as $\boldsymbol{L} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, where A is the adjacent matrix and $\boldsymbol{D}$ is the $(M + N) \times (M + N)$ diagonal degree matrix defined as $\boldsymbol{D}_{nn} = \sum_j \boldsymbol{A}_{nj}$.*

Our work focuses on recommendation based on implicit feedback, where we only observe whether a user has viewed or clicked an item. We denote $\mathcal{I}_i^+$ as the set of all items clicked by user i and denote $\mathcal{I}_i^-$ as the set of remaining items.

3.1 Binary Collaborative Filtering

Matrix factorization maps users and items onto a joint D-dimensional latent space, where user embedding matrix is represented by $\boldsymbol{P} = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_M]^{'} \in \mathbb{R}^{M \times D}$ and item embedding matrix is represented by $\boldsymbol{Q} = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_N]^{'} \in \mathbb{R}^{N \times D}$. However, binary collaborative filtering (Bin-CF) maps users and items onto a joint D-dimensional Hamming space. Denoting $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_M]^{'} \in \{-1, 1\}^{M \times D}$ and $\boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_M]^{'} \in \{-1, 1\}^{N \times D}$ as user and item's binary codes respectively, for implicit feedback, the Bin-CF

problem is formulated as follows:

$$\arg\min_{\boldsymbol{\Phi},\boldsymbol{\Psi}} \mathcal{L}_{Bin-CF} = \sum_{(i,j,j')\in\mathcal{D}} -\ln\sigma(\boldsymbol{p}_i^{\mathrm{T}}(\boldsymbol{q}_j - \boldsymbol{q}_{j'}))$$
$$\text{s.t.}\boldsymbol{\Phi} = H(\boldsymbol{P}), \boldsymbol{\Psi} = H(\boldsymbol{Q}) \quad (1)$$

where $H(\cdot)$ is a hash function:$\mathbb{R} \to \{-1, 1\}$

3.2 Binary-Continuous Equivalent Transformation

Let us consider the following generic binary program firstly,

$$\min f(\boldsymbol{x})$$
$$\text{s.t.}\boldsymbol{x} \in \{-1, 1\}^d \quad (2)$$

and a transformed problem,

$$\min f(\boldsymbol{x}) + \beta g(\boldsymbol{x})$$
$$\text{s.t.}\boldsymbol{x} \in [-1, 1]^d \quad (3)$$

where $g(\cdot)$:$\mathbb{R}^d \to \mathbb{R}$ is a penalty term for $f(\boldsymbol{x})$ and β is its penalty coefficient. [Giannessi and Tardella, 1998; Lucidi and Rinaldi, 2010] show that the above two problems are equivalent when certain conditions hold.

Lemma 1 *Denote $||\cdot||$ be a chosen norm. Suppose the following conditions hold:*

1) When $\boldsymbol{x} \in [-1, 1]^d$, $f(\boldsymbol{x})$ is bounded. In addition, there exists an open set $A \supset \{-1, 1\}^d$ and real positive number η, such that for $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in A$, the following Hölder condition is satisfied:

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq \eta||\boldsymbol{x}_1 - \boldsymbol{x}_2|| \quad (4)$$

2) $g(\cdot)$ satisfies:

(a) $g(\cdot)$ is continuous on $[-1, 1]^d$

(b) $\forall \boldsymbol{x} \in \{-1, 1\}^d, g(\boldsymbol{x}) = 0; \forall \boldsymbol{x} \in (-1, 1)^d, g(\boldsymbol{x}) > 0$

(c) $\forall \boldsymbol{y} \in \{-1, 1\}^d$, there exits a neighborhood $S(\boldsymbol{y})$ of $\boldsymbol{y}$ and a real positive number $\epsilon(\boldsymbol{y})$, such that:

$$\forall \boldsymbol{x} \in S(\boldsymbol{y}) \cap (-1, 1)^d, g(\boldsymbol{x}) \geq \epsilon(\boldsymbol{y})||\boldsymbol{x} - \boldsymbol{y}|| \quad (5)$$

Then there exits a real value η_0, such that $\forall \eta > \eta_0$, problem 2 and problem 3 are equivalent.

It can be verified that $g(\boldsymbol{x}) = |||\boldsymbol{x}| - \mathbf{1}||_F^2$ satisfies above conditions, and we adopt it as the penalty term.

4 Binarized Collaborative Filtering with Distilling Graph Convolutional Network

For binarized collaborative filtering for implicit feedback problem as shown in Eqn.1, there are three problems to solve. Firstly, the interaction information between users and items is extremely sparse. Secondly, a lot of information is lost during learning binary codes. Thirdly, binary optimization is general NP-hard, so we must adopt an efficient approximate method to solve it. We propose a novel framework-Binarized Collaborative Filtering with Distilling Graph Convolutional Network to deal with the aforementioned problems. Because GCN model can mine hidden connection information between users and items in user-item graph spectral domain, we train a GCN-based collaborative filtering model (GCN-CF) to solve the first problem. Then we utilize knowledge distillation to transfer the ranking information from GCN-CF into the binary model to make up for information loss. Finally, we propose a method to transform the binary optimization to a continuous optimization problem to solve the binary optimization problem.

4.1 GCN-based Collaborative Filtering

Following SpectralCF, our graph convolutional operation is shown as the following:

$$\begin{bmatrix}\boldsymbol{U}^{(k+1)}\\ \boldsymbol{V}^{(k+1)}\end{bmatrix} = \rho((\boldsymbol{I}_{M+N} + \boldsymbol{L})\begin{bmatrix}\boldsymbol{U}^{(k)}\\ \boldsymbol{V}^{(k)}\end{bmatrix}\boldsymbol{\Theta}^{(k)}) \quad (6)$$

where $\boldsymbol{L}$ is the Laplacian matrix of the bipartite graph $\mathcal{G}$. $\boldsymbol{I}_{M+N}$ is an identity matrix, $\rho(\cdot)$ is an activation function and $\boldsymbol{\Theta}^{(k)}$ is a layer-specific trainable filter parameter. The proposed convolution operation as shown in Eqn. (6) is denoted as $\text{sp}(\boldsymbol{X}; \boldsymbol{L}, \boldsymbol{\Theta})$.In this model, we set it as a two-layer GCN. According to Eqn. (1), similarity to matrix factorization (MF) methods, it does not take advantage of the user's own and the item's own high-order interaction, which limits the performance of GCN. Inspired by CrossNet [Wang *et al.*, 2017], we define the **cross operation**(cross_op) to fix the problem. The cross operation can be formulated as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k \boldsymbol{w}_k^T \boldsymbol{x}_k + \boldsymbol{x}_k \quad (7)$$

where $\boldsymbol{x}_k$ is a user or item's embedding vector and $\boldsymbol{w}_k$ is a parameter vector. The term $\boldsymbol{x}_k \boldsymbol{w}_k^T \boldsymbol{x}_k$ takes the place of the term $\boldsymbol{x}_0 \boldsymbol{w}_k^T \boldsymbol{x}_k$ in CrossNet, which leads to obtaining higher-order interactions than CrossNet when setting the same iterations. In addition, the time complexity of the proposed cross operation is still the same as CrossNet's. The improved GCN model can be vectorized as follows:

$$\begin{bmatrix}\boldsymbol{U}^{(1)}\\ \boldsymbol{V}^{(1)}\end{bmatrix} = \text{sp}(\begin{bmatrix}\boldsymbol{U}^{(0)}\\ \boldsymbol{V}^{(0)}\end{bmatrix}; \boldsymbol{L}, \Theta^{(0)}) \quad (8)$$

$$\begin{bmatrix}\boldsymbol{U}^{(2)}\\ \boldsymbol{V}^{(2)}\end{bmatrix} = \text{diag}((\begin{bmatrix}\boldsymbol{U}^{(1)}\\ \boldsymbol{V}^{(1)}\end{bmatrix} \circ \boldsymbol{W}_1)\mathbf{1})\begin{bmatrix}\boldsymbol{U}^{(1)}\\ \boldsymbol{V}^{(1)}\end{bmatrix} + \begin{bmatrix}\boldsymbol{U}^{(1)}\\ \boldsymbol{V}^{(1)}\end{bmatrix} \quad (9)$$

$$\begin{bmatrix}\boldsymbol{U}^{(3)}\\ \boldsymbol{V}^{(3)}\end{bmatrix} = \text{diag}((\begin{bmatrix}\boldsymbol{U}^{(2)}\\ \boldsymbol{V}^{(2)}\end{bmatrix} \circ \boldsymbol{W}_2)\mathbf{1})\begin{bmatrix}\boldsymbol{U}^{(2)}\\ \boldsymbol{V}^{(2)}\end{bmatrix} + \begin{bmatrix}\boldsymbol{U}^{(2)}\\ \boldsymbol{V}^{(2)}\end{bmatrix} \quad (10)$$

$$\begin{bmatrix}\boldsymbol{U}^{(4)}\\ \boldsymbol{V}^{(4)}\end{bmatrix} = \text{sp}(\begin{bmatrix}\boldsymbol{U}^{(3)}\\ \boldsymbol{V}^{(3)}\end{bmatrix}; \boldsymbol{L}, \Theta^{(1)}) \quad (11)$$

where "$\circ$" represents Hadamard product, "$\mathbf{1}$" is a column vector whose elements are all 1 and $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{(M+N)\times D}$ are weight matrices. Moreover, we add batch normalization [Ioffe and Szegedy, 2015] before Eqn.8 and Eqn.11.

In order to make full use of features from every layer of GCN, we follow SpectralCF and concatenate them into the final latent factors of users and items as:

$$\boldsymbol{U}^T = [\boldsymbol{U}^{(0)}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(4)}] \quad (12)$$
$$\boldsymbol{V}^T = [\boldsymbol{V}^{(0)}, \boldsymbol{V}^{(1)}, \boldsymbol{V}^{(4)}] \quad (13)$$

In terms of the loss function, we employ the popular and effective BPR loss [Rendle *et al.*, 2009]. In particular, given a

user matrix $\boldsymbol{U}$ and an item matrix $\boldsymbol{V}$ as shown in Eqn.12 and Eqn.13, the loss function of GCN-CF is given as

$$\mathcal{L}_{GCN-CF} = \sum_{(i,j,j')\in\mathcal{D}} -\ln\sigma(\boldsymbol{u}_i^{\mathrm{T}}(\boldsymbol{v}_j - \boldsymbol{v}_{j'})) + \lambda(||\boldsymbol{U}||_F^2 + ||\boldsymbol{V}||_F^2) \quad (14)$$

where $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ denote ith and jth rows of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively; λ is the regularization coefficient. Negative sample $j^{'}$ is sampled from $\mathcal{I}_i^-$ randomly and the training data D is generated as $\mathcal{D} = \{(i, j, j^{'})|i \in \mathcal{U} \wedge j \in \mathcal{I}_i^+ \wedge j^{'} \in \mathcal{I}_i^-\}$.

Distilling GCN into Binarized Collaborative Filtering

In this model, we distill the ranking information in GCN-CF model and transfer it to a simple binary collaborative filtering model via a mixed objective function. The model is denoted by DGCN-BinCF for short. The *key motivation* of the distillation in DGCN-BinCF is two-fold. On one hand, we consider the distribution of positive (negative) samples in the binary model should be close to that in the GCN-CF. On the other hand, the differences between positive and negative samples should become far enough in DGCN-BinCF. However, because the BPR model can guarantee that positive samples have higher scores than negative samples', negative samples are assigned much lower probability than positive samples' if we consider the distribution of both positive and negative samples at the same time. Thus, we consider positive and negative samples in GCN-CF separately.

Specifically, to distill the ranking information in GCN-CF, we hope the positive (negative) items of one user have the approximately same order in binary model and continuous model. For instance, if user u's preference for the three items i, j, k is ranked as $[i, j, k]$ in GCN-CF model, we hope that the rank keeps $[i, j, k]$ in the binary model. According to ListNet [Cao *et al.*, 2007], we can characterize sorting information in the following ways

$$\mathcal{L}_{rank} = \sum_{(i,j)\in\mathcal{D}^+} -\frac{\exp(\frac{\boldsymbol{u}_i^T\boldsymbol{v}_j}{T})}{\sum_{j\in\mathcal{D}^+}\exp(\frac{\boldsymbol{u}_i^T\boldsymbol{v}_j}{T})}\log(\frac{\exp(\frac{\boldsymbol{p}_i^T\boldsymbol{q}_j}{T})}{\sum_{j\in\mathcal{D}^+}\exp(\frac{\boldsymbol{p}_i^T\boldsymbol{q}_j}{T})}) + \sum_{(i,j)\in\mathcal{D}^-} -\frac{\exp(\frac{\boldsymbol{u}_i^T\boldsymbol{v}_j}{T})}{\sum_{j\in\mathcal{D}^-}\exp(\frac{\boldsymbol{u}_i^T\boldsymbol{v}_j}{T})}\log(\frac{\exp(\frac{\boldsymbol{p}_i^T\boldsymbol{q}_j}{T})}{\sum_{j\in\mathcal{D}^-}\exp(\frac{\boldsymbol{p}_i^T\boldsymbol{q}_j}{T})}) \quad (15)$$

where $\mathcal{D}^+ = \{(i, j)|i \in \mathcal{U} \wedge j \in \mathcal{I}_i^+\}$, $\mathcal{D}^- = \{(i, j^{'})|i \in \mathcal{U} \wedge j^{'} \in \mathcal{I}_i^-\}$ and T is the temperature parameter. In Eqn.15, we convert the items' score list to probability distributions via softmax function, and utilize cross entropy for penalizing the discrepancy. According to [Hinton *et al.*, 2015], combining $\mathcal{L}_{Bin-CF}$ with $\mathcal{L}_{rank}$ as a multi-task learing problem can transfer the ranking knowledge to the binary model. It's worth mentioning that since the magnitudes of the gradients produced by the $\mathcal{L}_{rank}$ scale as $1/T^2$, it is necessary to multiply them by T^2 when mixing $\mathcal{L}_{rank}$ and $\mathcal{L}_{Bin-CF}$. So the loss function of DGCN-BinCF is formulated as

$$\mathcal{L}_{DGCN-BinCF} = \mathcal{L}_{Bin-CF} + \alpha T^2 \mathcal{L}_{rank} \quad (16)$$

Algorithm 1: DGCN-BinCF Algorithm

Input: Bipartite graph:$\mathcal{G}$;dimension of latent factor in GCN-CF:D;dimension of latent factor in DGCN-BinCF:$d = 3D$;Laplacian matrix:$\boldsymbol{L}$;temperature:T, τ;coefficient:$\lambda, \alpha, \eta, \nu$

Initialize GCN-CF model;
repeat
  Randomly sampling from unobserved items to generate training set $\mathcal{D}$;
  Forward propagation according to Eqn.8 $\sim$ 11;
  Concatenate $\boldsymbol{U}$ and $\boldsymbol{V}$ via Eqn.12 and 13;
  Update parameters of GCN-CF model by Adam optimization;
until *Convergence*;
Initialize DGCN-BinCF model;
repeat
  Randomly sampling from unobserved items to generate training set $\mathcal{D}$;
  Forward propagation according to Eqn.18;
  Update parameters of DGCN-BinCF model by Adam optimization;
until *Convergence*;

$\mathcal{L}_{DGCN-BinCF}$ is denoted as $\mathcal{L}(\boldsymbol{P}, \boldsymbol{Q}; \boldsymbol{U}, \boldsymbol{V}, \alpha, T)$ for short. $\boldsymbol{P}$ and $\boldsymbol{Q}$ are user and item embedding matrices in DGCN-BinCF respectively, and $\boldsymbol{U}$, $\boldsymbol{V}$ are trained user and item embedding matrices of GCN-CF. The loss $\mathcal{L}_{rank}$ encodes the first motivation and the loss $\mathcal{L}_{Bin-CF}$ encodes the second motivation.

For the binary optimization problem, it is a direct method to use $\tanh(x/t)$ to approximate sign function, where t is a small temperature. But [Li *et al.*, 2018b] points that setting a small temperature will harm the optimization process. [Courbariaux *et al.*, 2015] mentions that generating binary codes stochastically is a finer and more correct averaging process than generating binary codes via sign function. Hence, we generate binary codes via sampling from the Bernoulli distribution. More specifically, given $\boldsymbol{x} \in (-1, 1)^d$, its corresponding binary code is $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is a random variable which only can be $\mathbf{1} - \boldsymbol{x}$ and $-\mathbf{1} - \boldsymbol{x}$, and $P(\boldsymbol{\varepsilon}_i = 1 - \boldsymbol{x}_i) = \mathrm{sigmoid}(\boldsymbol{x}_i/\tau)$, $P(\boldsymbol{\varepsilon}_i = -1 - \boldsymbol{x}_i) = 1 - \mathrm{sigmoid}(\boldsymbol{x}_i/\tau)$. Here, τ is temperature parameter. To force the noise to be small, we add the expectation of noise as a penalty term. Therefore, the DGCN-BinCF can be transformed into the following optimization problem:

$$\min \mathcal{L}(\tanh(\boldsymbol{P}) + \boldsymbol{\varepsilon}_{\boldsymbol{P}}, \tanh(\boldsymbol{Q}) + \boldsymbol{\varepsilon}_{\boldsymbol{Q}}; \boldsymbol{U}, \boldsymbol{V}, \alpha, \mathrm{T}) + \nu(\mathbb{E}(||\boldsymbol{\varepsilon}_{\boldsymbol{P}}||_F^2) + \mathbb{E}(||\boldsymbol{\varepsilon}_{\boldsymbol{Q}}||_F^2)) \quad (17)$$

where $(\boldsymbol{\varepsilon}_{\boldsymbol{P}})_{ij} \sim Bernoulli(\tanh(\boldsymbol{P}_{ij}))$, $(\boldsymbol{\varepsilon}_{\boldsymbol{Q}})_{ij} \sim Bernoulli(\tanh(\boldsymbol{Q}_{ij}))$. We use the **tanh** function to bound the value of $\boldsymbol{P}, \boldsymbol{Q}$ between -1 and 1.

According to Lemma 1, Eqn 17 can be rewritten as

$$\min \mathcal{L}(\tanh(\boldsymbol{P}), \tanh(\boldsymbol{Q}); \boldsymbol{U}, \boldsymbol{V}, \alpha, \mathrm{T}) + \nu(\mathbb{E}(||\boldsymbol{\varepsilon}_{\boldsymbol{P}}||_F^2) + \mathbb{E}(||\boldsymbol{\varepsilon}_{\boldsymbol{Q}}||_F^2)) + \beta\Big(\sum_i g\big(\tanh(\boldsymbol{p}_i)\big) + \sum_j g\big(\tanh(\boldsymbol{q}_j)\big)\Big) \quad (18)$$

Dataset	#User	#Item	#Rating	Density
MovieLens1M	6,022	3,043	995,154	5.43%
MovieLens10M	69,878	10,681	10,000,054	1.34%
Yelp	9,235	7,353	423,354	0.62%

Table 1: Statistics of datasets

Here we adopt $g(\boldsymbol{p}_i) = |||\boldsymbol{p}_i| - \mathbf{1}||_2^2$ which can be validated satisfying the conditions as the penalty term. Eqn.18 can be optimized by any gradient-based optimization methods directly. The whole training processing is summarized in Algorithm1.

5 Experiments

In this section, we evaluate our proposed DGCN-BinCF framework with the aim of answering the following research questions.

1. Does the recommendation performance of the proposed DGCN-BinCF framework outperforms the state-of-the-art hashing-based recommendation methods?
2. Whether our proposed GCN-CF is effective?
3. Whether distilling ranking information helps learning binary model?
4. Whether this proposed framework can converge well?

We introduce the experimental settings firstly and then answer the above questions in following sections.

5.1 Experiment Settings

Dataset

We use three public real datasets including *MovieLens1M*, *MovieLens10M* and *Yelp* to evaluate the proposed algorithm. Because the three datasets are explicit feedback data, to convert them into implicit feedback data, we set all ratings as positive samples. In addition, due to the extreme sparsity of them, we then filter users who have less than 20 ratings and remove items that are rated by less than 20 users. Table 1 summaries the filtered datasets. For each user, we sampled randomly 50% positive samples as training and the remaining as test. We repeated five random splits and reported the averaged results.

Comparison Methods

To evaluate the performance of DGCN-BinCF for hashing-based recommender systems, we compare DGCN-BinCF with 3 very popular and state-of-art methods: DCF, BCCF and PPH. DCF solves the binary optimization problem directly via bit-wise optimization. BCCF and PPH are two-stage methods.

To measure the effectiveness of the improved GCN model, we compare GCN-CF with SpectralCF. And we compare DGCN-BinCF with the binary model $\mathcal{L}_{Bin-CF}$. To show the role of KD loss in binary optimization, $\mathcal{L}_{Bin-CF}$ is optimized by our proposed relaxation method as well.

	Recall@100	MAP@100	NDCG@100
DCF	0.0416	0.0101	0.0558
BCCF	0.1234	0.0720	0.1724
PPH	0.0277	0.0027	0.0249
DGCN-BinCF	0.3059	0.1187	0.3061

Table 2: Item Recommendation Results(MovieLens1M)

	Recall@100	MAP@100	NDCG@100
DCF	0.0791	0.0180	0.0809
BCCF	0.0789	0.0267	0.0869
PPH	0.0978	0.0123	0.0695
DGCN-BinCF	0.2405	0.0537	0.1895

Table 3: Item Recommendation Results(MovieLens10M)

	Recall@100	MAP@100	NDCG@100
DCF	0.0661	0.0053	0.0382
BCCF	0.0966	0.0122	0.0658
PPH	0.0627	0.0051	0.0361
DGCN-BinCF	0.1738	0.0192	0.1008

Table 4: Item Recommendation Results(Yelp)

Evaluation Metric

To evaluate the recommendation system performance, we choose four widely used ranking-based metric: (1) NDCG (Normalized Discounted Cumulative Gain), (2) Recall, , and (3) MAP (Mean Average Precision). We predicted the top-K preferred items from test set for each user in our experiments.

Parameter Settings

In our experiments, we set the regularization coefficient $\lambda = 0.001$ in GCN-CF in all dataset. For DGCN-BinCF, we set temperature $T = 1$, $\tau = 0.2$, the weight $\alpha = 10$, and the penalty coefficients $\beta = 0.001$, $\nu = 0.001$ in the three datasets. In addition, we set the dimension of users and items' latent factor of GCN-CF 16 in *MovieLens10M* and 64 in the other two datasets. The learned matrices $\boldsymbol{U}$,$\boldsymbol{V}$ in GCN-CF are used as the initialization of $\boldsymbol{P}$,$\boldsymbol{Q}$ in DGCN-BinCF. All parameters of SpectralCF are set according to [Zheng *et al.*, 2018].

Besides, for DCF, BCCF and PPH, we held-out evaluation means on splits of training data randomly to tune the optimal hyper-parmenters via grid search. α and β in DCF are tuned among the set $\{1e-4, 1e-3, \cdots, 1e1\}$. λ in BCCF is tuned among the set $\{0.01, 0.03, \cdots, 0.09\}$ and λ in PPH is tuned among the set $\{0.01, 0.5, 1, 2, 4, 8, 16\}$.

5.2 Comparison with Baselines

Although hashing-based recommendation has significant advantages of both time and storage, it often incurs low accuracy recommendation because binary codes have limited

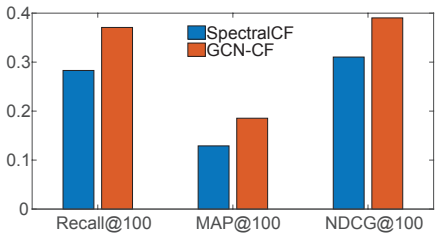

Figure 1: The performance of SpectralCF and GCN-CF

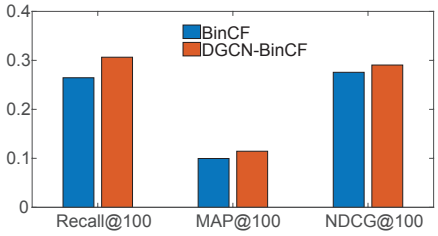

Figure 2: The comparison between BinCF and DGCN-BinCF

representation ability and lose a lot of information compared with real-valued recommender systems. DGCN-BinCF is to improve the accuracy of recommendation.

In this part we will answer the first question. We compare the recommendation accuracy of DGCN-BinCF with three state-of-art binary recommendation methods including DCF, BCCF and PPH on the three datasets. Table 2, Table 3 and Table 4 summary the results.

The three tables show that DGCN-BinCF has much better performance than all baselines on the three datasets. This is because we train the improved GCN model GCN-CF firstly to discover the deep interactions between users and items and then transfer the ranking information to a binary model, DGCN-BinCF loses less information than baseline models. In addition, the binary optimization problem is optimized directly by the proposed penalty terms, which leads to less quantization loss. Therefore, DGCN-BinCF has great advantages over DCF, BCCF and PPH.

5.3 The Effectiveness of GCN-CF

It is mentioned that SpectralCF did not consider aggregating users and items' own high-order feature, which may limit its representation ability. In this part, we will answer the second question.

We implement the experiment in MovieLens1M dataset. We utilize three metrics to evaluate the performance of SpectralCF and GCN-CF respectively. Figure 1 shows the results. In two histograms, the orange column is the performance of GCN-CF and the blue one represents the results of SpectralCF. From the histograms, it is clear to observe that GCN-CF has great improvement (over 20%) for every metric compared with SpectralCF, which shows the effectiveness of GCN-CF.

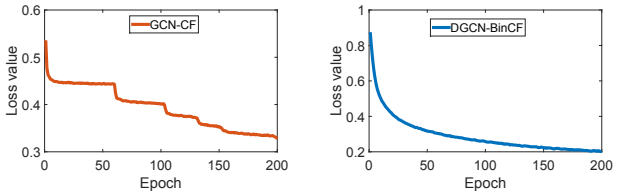

Figure 3: The left one is the loss-epoch figure of GCN-CF model;the right one is the loss-epoch figure of DGCN-BinCF model.

5.4 The Effectiveness of Distillation

Because lots of useful information loses during learning the binary representation, it is vital to utilize the ranking information from the trained GCN-CF model as supplements for learning discrete codes. In this part, we investigate the role of ranking information for binary optimization. To implement the experiment, we consider optimizing Eqn.1 directly by adding the proposed penalty term. In the other word, we set $\alpha = 0$ in $\mathcal{L}_{DGCN-BinCF}$, and compare its results with DGCN-BinCF. We test the two methods in the MovieLens1M dataset and evaluate them via the four ranking metrics.

Figure 2 reports the comparison results. The blue bar represents BinCF and the orange bar means DGCN-BinCF model. The two histograms show that for all evaluation indicators, DGCN-BinCF outperforms BinCF by 10%. Thus we conclude that the distillation method helps the model learn high quality binary representation.

5.5 Convergence

In this section, we will answer the forth question. Because deep models and discrete optimization may diverge, we test the convergence of GCN-CF and DGCN-BinCF model.

To test the convergence of our proposed model GCN-CF and DGCN-BinCF, we implement the experiment on *MovieLens1M*. We record the value of Eqn.14 and Eqn.18 with the change of epoch respectively. In this experiment, we set the maximum number of iterations 200. Figure 3 shows the convergence of two models. It is observed that loss value of GCN-CF decreases and the DGCN-BinCF converges greatly.

6 Conclusion

In this paper, we propose a hash-based method DGCN-BinCF to accelerate implicit feedback recommendation. Because implicit feedback lacks negative samples and learning binary codes loses a lot of information, we train the model GCN-CF, which aggregates users' and items' own high-order feature, to mine rich connection information, and then distill the ranking information from GCN-CF into the binary model. In addition, we propose a method utilizing penalty terms to learning binary codes based on gradient descent directly. The experiments on three real-world datasets show the great superiority of our framework.

Acknowledgements

The work was supported in part by grants from the National Natural Science Foundation of China (Grant No. U1605251, 61832017, 61631005 and 61502077).

References

[Berg *et al.*, 2017] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[Cao *et al.*, 2007] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of ICML'07*, pages 129–136. ACM, 2007.

[Chen *et al.*, 2018] Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. In *Proceedings of AAAI'18*, 2018.

[Courbariaux *et al.*, 2015] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proceedings of NeurIPS'15*, pages 3123–3131, 2015.

[Dai *et al.*, 2016] Qi Dai, Jianguo Li, Jingdong Wang, and Yu-Gang Jiang. Binary optimized hashing. In *Proceedings of MM'16*, pages 1247–1256. ACM, 2016.

[Das *et al.*, 2007] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of WWW'07*, pages 271–280. ACM, 2007.

[Datar *et al.*, 2004] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[Gao *et al.*, 2018] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of KDD'18*, pages 1416–1424. ACM, 2018.

[Giannessi and Tardella, 1998] Franco Giannessi and Fabio Tardella. Connections between nonlinear programming and discrete optimization. In *Handbook of combinatorial optimization*, pages 149–188. Springer, 1998.

[Håstad, 2001] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[Karatzoglou *et al.*, 2010] Alexandros Karatzoglou, Alex Smola, and Markus Weimer. Collaborative filtering on a budget. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 389–396, 2010.

[Li *et al.*, 2018a] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proceedings of KDD'18*, pages 1734–1743. ACM, 2018.

[Li *et al.*, 2018b] Zhuohan Li, Di He, Fei Tian, Wei Chen, Tao Qin, Liwei Wang, and Tie-Yan Liu. Towards binary-valued gates for robust lstm training. *arXiv preprint arXiv:1806.02988*, 2018.

[Lian *et al.*, 2016] Defu Lian, Yuyang Ye, Wenya Zhu, Qi Liu, Xing Xie, and Hui Xiong. Mutual reinforcement of academic performance prediction and library book recommendation. In *Proceedings of ICDM'16*, pages 1023–1028. IEEE, 2016.

[Liu *et al.*, 2011] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. Personalized travel package recommendation. In *Proceedings of ICDM'11*, pages 407–416. IEEE, 2011.

[Liu *et al.*, 2014] Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. Collaborative hashing. In *Proceedings of CVPR'14*, pages 2139–2146, 2014.

[Lucidi and Rinaldi, 2010] Stefano Lucidi and Francesco Rinaldi. Exact penalty functions for nonlinear integer programming problems. *Journal of optimization theory and applications*, 145(3):479–488, 2010.

[Monti *et al.*, 2017] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707, 2017.

[Muja and Lowe, 2009] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[Pan *et al.*, 2008] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Proceedings of ICDM'08*, pages 502–511. IEEE, 2008.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI'09*, pages 452–461. AUAI Press, 2009.

[Tang and Wang, 2018] Jiaxi Tang and Ke Wang. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of KDD'18*, pages 2289–2298. ACM, 2018.

[Wang *et al.*, 2012] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):2393–2406, 2012.

[Wang *et al.*, 2017] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, page 12. ACM, 2017.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of KDD'18*, pages 974–983. ACM, 2018.

[Zhang *et al.*, 2014] Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. Preference preserving hashing for efficient recommendation. In *Proceedings of SIGIR'14*, pages 183–192. ACM, 2014.

[Zhang *et al.*, 2016] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. Discrete collaborative filtering. In *Proceedings of SIGIR'16*, pages 325–334. ACM, 2016.

[Zheng *et al.*, 2018] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. Spectral collaborative filtering. In *Proceedings of RecSys'18*, pages 311–319. ACM, 2018.

[Zhou and Zha, 2012] Ke Zhou and Hongyuan Zha. Learning binary codes for collaborative filtering. In *Proceedings of KDD'18*, pages 498–506. ACM, 2012.