HCFRec: Hash Collaborative Filtering via Normalized Flow with Structural Consensus for Efficient Recommendation

Fan Wang, **Weiming Liu**, **Chaochao Chen**, **Mengying Zhu** and **Xiaolin Zheng***
College of Computer Science and Technology, Zhejiang University, China
fanwang1997@hotmail.com, {21831010, zjuccc, mengyingzhu, xlzheng}@zju.edu.cn

Abstract

The ever-increasing data scale of user-item interactions makes it challenging for an effective and efficient recommender system. Recently, hash-based collaborative filtering (Hash-CF) approaches employ efficient Hamming distance of learned binary representations of users and items to accelerate recommendations. However, Hash-CF often faces two challenging problems, i.e., *optimization on discrete representations* and *preserving semantic information in learned representations*. To address the above two challenges, we propose HCFRec, a novel Hash-CF approach for effective and efficient recommendations. Specifically, HCFRec not only innovatively introduces normalized flow to learn the optimal hash code by efficiently fitting a proposed *approximate mixture multivariate normal distribution*, a continuous but approximately discrete distribution, but also deploys a cluster consistency preserving mechanism to preserve the semantic structure in representations for more accurate recommendations. Extensive experiments conducted on six real-world datasets demonstrate the superiority of our HCFRec compared to the state-of-art methods in terms of effectiveness and efficiency.

1 Introduction

Recommender System (RS) has recently become a crucial tool to alleviate information overload in many areas with rich data, including but not limited to e-commerce, education, finance, and health [Zhu *et al.*, 2021]. As a pivotal technology of RS, Collaborative Filtering (CF) has received thrilling success owing to its inherent domain-independent and easy-to-explain properties. Specifically, CF attempts to learn representations of users and items from their interactive information for the subsequent user preference prediction and item recommendation[Chen *et al.*, 2022]. However, with the explosive growth of users and items, CF based recommendation methods often suffer from high time and space costs [Qi *et al.*, 2021a].

Fortunately, hash-based CF (Hash-CF) approaches [Chen *et al.*, 2018] have been proven to have a good ability to compress data and accelerate computation for recommendations with billion-scale users and items [Shan *et al.*, 2018]. In large-scale recommendation scenarios, Hash-CF takes effect by encoding high dimensional real-valued vectors into compact one-hot codes (*hash representations*), such that: (1) bit-wise operations (e.g., XOR) instead of real-valued calculations for preference inference can dramatically accelerate recommendations; (2) bit-wise representations often achieve a $64\times$ storage compression rate compared to real-valued representations. Therefore, Hash-CF enables a lighter and more efficient recommendation model even massive data are involved in decision-makings.

However, existing Hash-CF approaches often face two challenges existing in the learned hash representations. Firstly, **CH1:** *how to implement optimization on discrete hash representations?* The hash representation is usually obtained through the sign function, and optimizing such a representation will lead to a challenging mixed-binary-integer optimization problem [Wang *et al.*, 2018], which is NP-hard. A promising solution is to replace the sign function with a continuous relaxation (e.g., $\tanh$ function) to learn deterministic hash representations in an end-to-end manner, which, however, is not robust due to a lack of noise tolerance consideration. Fortunately, Variational Autoencoder (VAE) [Kingma and Welling, 2014] with its probabilistic nature can model features as distributions to accommodate much uncertainty or noisy in data, so as to implement robust recommendation [Liang *et al.*, 2018]. However, for the Hash-CF task, features need to be modeled as latent discrete Bernoulli distributions to generate hash representations. Such distributions with discrete nature make the optimization on hash representations more difficult. Secondly, **CH2:** *how to preserve semantic information in discrete representations?*A hash representation has intrinsically limited representation ability, as it carries less semantic information than a real-valued representation. Although the existing Hash-CF approaches [Zhang *et al.*, 2016] try to control quantization loss to reduce the difference between real-valued and hash representations, they fail to preserve the semantic structure consistency between them. The two representations without structural consensus drop much semantic information that is crucial to accurate recommendations.

*Corresponding author

In light of the above two challenges, we propose a novel Hash-CF recommendation approach, i.e., HCFRec, which generates compact yet informative hash representations for effective and efficient recommendations. The proposal is comprised of two major components based on VAE framework for robust generalization capability. Specifically, for **CH1**, *hash representation generation component* first models user (item) features as user (item) real-valued representations that obey a simple prior normal distribution. Considering the difficulty in optimizing a discrete distribution, we innovatively propose an *approximate mixture multivariate normal distribution*, a continuous but approximately discrete distribution, and introduce normalized flow [Rezende and Mohamed, 2015] to deploy reversible transformation functions from the simple normal distribution to the complex and approximately discrete distribution. Normalized flow can efficiently implement reversible distribution transformation from a simple continuous distribution to any complex continuous distribution with same dimensions. To our best knowledge, this is the first attempt to introduce normalized flow for binary optimization. Finally, hash representations are generated by binarization operations. For **CH2**, *cluster consistency preserving component* first respectively clusters the real-valued representations and the hash distribution both generated in the first component. Subsequently, a loss function is designed for the two representations to alternately learn semantic information with structural consensus.

We summarize our main contributions as follows: (1) We propose a novel Hash-CF approach implemented on a variational framework for effective and efficient recommendation. (2) Normalized flow is first introduced to learn the optimal hash codes by efficiently fitting our proposed *approximate mixture multivariate normal distribution*, which makes it possible for efficient optimization on discrete distribution. (3) We innovatively develop a cluster structure preserving mechanism to retain cluster consensus between real-valued and hash representations for more accurate recommendations. (4) Extensive experiments conducted on six real-world datasets demonstrate the superiority of our HCFRec approaches over state-of-the-art competitive ones.

2 Related Work

2.1 Hash-CF Methods

Due to the intrinsic characteristics of high efficiency and low storage cost, Hash-CF has recently gained ever-increasing attentions of researchers, including two kinds of approaches. First, the "two-stage" approaches learn hash representations through a continuous representation learning stage followed by a binary quantization stage [Karatzoglou *et al.*, 2010; Zhang *et al.*, 2014; Zhang *et al.*, 2017b], which, however, produces considerable quantization errors. Second, the "end-to-end" approaches try to optimize hash representations directly. Some researchers model user and item features as deterministic representations [Zhang *et al.*, 2016][Zhang *et al.*, 2017a][Liu *et al.*, 2019]. However, these approaches lack noise tolerance. For more generalization, some researchers recruit probability-based VAE to model user/item features as discrete Bernoulli distributions [Hansen *et al.*, 2020]. However, discrete latent space makes it challenging for hash codes to be optimized effectively and efficiently.

2.2 VAE-based CF Methods

VAE has recently achieved strong performance improvements on CF [Liang *et al.*, 2018]. VAE's strong generative ability even in sparse settings is mainly because it can model representations as a distribution rather than a deterministic vector to account for much uncertainty in the latent space. [Lee *et al.*, 2017] incorporates auxiliary information into VAE for CF. [Karamanolakis *et al.*, 2018] also takes side information into account, which learns user representations in multimodal latent space for better recommendation. RecVAE [Karamanolakis *et al.*, 2018] introduces some regularization techniques into VAE to improve recommendation performance. However, despite remarkable improvements are achieve in these works, they only learn user representations but ignore valuable item information. In this situation, [Truong *et al.*, 2021] models a Bilateral VAE framework to learn both user and item representations for more robust recommendation.

3 Methodology

The data we intend to learn from is a user-item interaction matrix $\boldsymbol{R} \in \mathbb{R}^{N_U \times N_V}$, where N_U and N_V denote the number of users and items, respectively. The i-th row of $\boldsymbol{R}$ is formulated as $\boldsymbol{r_{i,*}} \in \mathbb{R}^{N_V}$ while the j-th column of $\boldsymbol{R}$ is formulated as $\boldsymbol{r_{*,j}} \in \mathbb{R}^{N_U}$, representing user feature and item feature, respectively. Moreover, E_U and E_V are respectively user encoder and item encoder, $\boldsymbol{b^U} \in \{-1, 1\}^D$ and $\boldsymbol{b^V} \in \{-1, 1\}^D$ respectively indicate user hash representations and item hash representations. The main problem we need to solve is: *Given $\boldsymbol{r_{i,*}}$ and $\boldsymbol{r_{*,j}}$ from matrix $\boldsymbol{R}$, a model is needed to learn hash representations $\boldsymbol{b^U}$ and $\boldsymbol{b^V}$ for effective and efficient recommendations.*

3.1 Overview of HCFRec

Our HCFRec model is mainly based on a dual VAE framework comprised of *hash representation generation component* and *cluster consistency preserving component*, as illustrated in Figure 1. Details of each component are as follows:

(1) Hash Representation Generation Component. Due to the close correlation with the VAE framework, this component is embedded into the VAE for introduction here. Concretely, we first employ VAE framework to model user (item) features as real-valued representations that obey prior normal distribution. Normalized flow is subsequently deployed to achieve pre-hash representation that obeys a continuous but approximately discrete distribution. Finally, sign function on this distribution is recruited to output hash representations.

(2) Cluster Consistency Preserving Component. We cluster the embedded real-valued representations and the generated hash representations, and alternately learn cluster-consistent semantic information from each other to ensure the semantic structure restoration ability of hash representations. Finally, HCFRec uses the enhanced hash representations for recommendations.

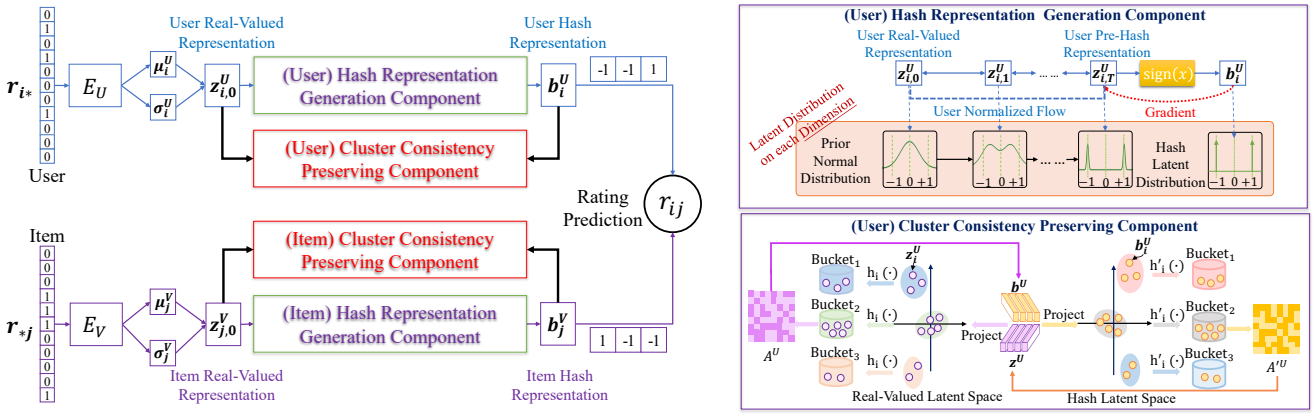


Figure 1: The overview of HCFRec. The left is the model framework based on dual VAE, comprised of hash representation generation component and cluster consistency preserving component. The right is the details of the two components, where we employ user side as a specific example.

3.2 Hash Representation Generation Component

Modelling. For a recommendation task, the objective we seek to maximize is the likelihood of a rating, i.e., $p(r_{i,j})$. Formally, we determine the likelihood $p(r_{i,j})$ in Eq.(1) by introducing binary latent representations $\mathbf{b^U}$ and $\mathbf{b^V}$ as conditions.

$$p(r_{i,j}) = \sum_{\mathbf{b_i^U}, \mathbf{b_j^V} \in \{-1,1\}^D} p(r_{i,j}|\mathbf{b_i^U}, \mathbf{b_j^V}) p(\mathbf{b_i^U}) p(\mathbf{b_j^V}) \quad (1)$$

However, such maximization is intractable, thus we need to introduce variational inference [Jordan *et al.*, 1999]. Variational inference devotes to approximate both posteriors $q_\phi(\mathbf{b_i^U}|\mathbf{r_{i,*}})$ and $q_\psi(\mathbf{b_j^V}|\mathbf{r_{*,j}})$ to surrogate the true intractable posterior $p(\mathbf{b_i^U}|\mathbf{r_{i,*}})$ and $p(\mathbf{b_j^V}|\mathbf{r_{*,j}})$, where ϕ and ψ are parameters on neural networks respectively for user and item. Formally, variational inference in this work seeks to simultaneously minimize the KL divergence $\mathrm{KL}(q_\phi(\mathbf{b_i^U}|\mathbf{r_{i,*}})\|p(\mathbf{b_i^U}|\mathbf{r_{i,*}})$ and $\mathrm{KL}(q_\psi(\mathbf{b_j^V}|\mathbf{r_{*,j}}\|p(\mathbf{b_j^V}|\mathbf{r_{*,j}})$ to achieve the optimal ϕ and ψ. During the learning process with the above variational inference, an Evidence Lower BOund can be drawn in Eq.(2). Therefore, we can use ELBO maximization as a proxy to indirectly maximize the log-likelihood function.

$$\begin{aligned} \log p(r_{i,j}) \geq & \sum_{i,j} \mathbb{E}_{q_\phi(\mathbf{b_i^U}|\mathbf{r_{i,*}})} \mathbb{E}_{q_\psi(\mathbf{b_j^V}|\mathbf{r_{*,j}})} [\log\ p(r_{i,j}|\mathbf{b_i^U}, \mathbf{b_j^V})] \\ & - \sum_i \mathrm{KL}(\mathrm{q}_\phi(\mathbf{b_i^U}|\mathbf{r_{i,*}})\|\mathrm{p}(\mathbf{b_i^U})) - \sum_\mathrm{j} \mathrm{KL}(\mathrm{q}_\psi(\mathbf{b_j^V}|\mathbf{r_{*,j}})\|\mathrm{p}(\mathbf{b_j^V})) \end{aligned} \quad (2)$$

where the first term in ELBO indicates the reconstruction error that measures the likelihood of reconstructing the observed rating data, the two KL terms are regularizers that constrain the form of the two approximate posteriors.

Calculation. In this part, user side and item side are treated symmetrically. Due to the limited space, this part only takes user side as a specific example for illustration. Firstly, we adopt $\boldsymbol{E}_U$ to generate user mean and variance as $[\boldsymbol{\mu}_i^U, (\boldsymbol{\sigma}_i^U)^2] = \boldsymbol{E}_U(\boldsymbol{r}_{i*})$. Then we adopt the reparametric method to obtain the user real-valued representation as $\boldsymbol{z}_{i,0}^U = \boldsymbol{\mu}_i^U + \epsilon_i^U \boldsymbol{\sigma}_i^U$ where $z_{i,0}^U$ is a D-dimensional vector. In order to enhance the model generalization, we align the real-valued latent space to the standard normal distribution $\mathcal{N}(0, \boldsymbol{I})$ with the KL-divergence constraint as:

$$\begin{aligned} \min \mathcal{L}_{Align}^U &= \min \mathrm{KL}(\mathcal{N}(\boldsymbol{\mu}_i^U, (\boldsymbol{\sigma}_i^U)^2)\|\mathcal{N}(0, \boldsymbol{I})) \\ &= \frac{1}{2} \sum_{i=1}^{N} \left[(\boldsymbol{\mu}_i^U)^2 + (\boldsymbol{\sigma}_i^U)^2 - \log(\boldsymbol{\sigma}_i^U)^2 - 1 \right] \end{aligned} \quad (3)$$

After that we adopt the normalized flow to generate the $\boldsymbol{z}_{i,T}^U$ through several layers with probability estimation as:

$$\log q_\phi(\boldsymbol{z}_{i,T}^U|_d) = \log q_\phi(\boldsymbol{z}_{i,0}^U|_d) - \sum_{i=1}^{T-1} \log \left| \det \frac{d\boldsymbol{z}_{i,t+1}^U|_d}{d\boldsymbol{z}_{i,t}^U|_d} \right| \quad (4)$$

where $\boldsymbol{z}_{i,t}^U + \boldsymbol{u}_t^U \chi\left((\boldsymbol{w}_t^U)^\top \boldsymbol{z}_{i,t}^U + \boldsymbol{a}_t^U\right) = \boldsymbol{z}_{i,t+1}^U \in \mathbb{R}^D$ and $\chi(\cdot)$ denotes the sigmoid activation function. $\boldsymbol{u}_t^U$, $\boldsymbol{w}_t^U$ and $\boldsymbol{a}_t^U$ are the trainable network parameters at the t-th layer. Here, it is worth noting that the reversible distribution transformation in Eq.(4) is only deployed on the d-th dimension of the entire D-dimensional representation. Finally, we adopt $\mathrm{sign}(\cdot)$ to achieve the user hash representation as:

$$\boldsymbol{b}_i^U = \mathrm{sign}(\boldsymbol{z}_{i,T}^U) \quad (5)$$

Meanwhile we prefer that the latent hash representation conforms to the Bernoulli distribution as $p(\boldsymbol{b}_i|_d = 1) = p(\boldsymbol{b}_i|_d = -1) = \frac{1}{2}$ on the d-th dimension. However, the Bernoulli distribution is discrete which makes it much more difficult to be optimized. Therefore, we first innovatively propose the *approximate mixture multivariate normal distribution* represented as:

$$p(\boldsymbol{b}_i^U|_d) \approx p(\boldsymbol{z}_{i,T}^U|_d) = \frac{1}{2} \left[\mathcal{N}(\mathbf{1}, \boldsymbol{\Sigma}(b_i^U|_d)) + \mathcal{N}(-\mathbf{1}, \boldsymbol{\Sigma}(b_i^U|_d)) \right] \quad (6)$$

where $\boldsymbol{\Sigma}(b_i|_d) = \gamma \boldsymbol{I}$ denotes the covariance matrix of corresponding normal distribution. Here, γ is set to 0.015 in our experiments. Therefore, we define the loss function corresponding to the KL term in Eq. (2) as:

$$\begin{aligned} \mathcal{L}_{\mathrm{KL}}^U &= \sum_i \mathrm{KL}(q_\phi(\mathbf{b_i^U}|\mathbf{r_{i,*}})\|p(\mathbf{b_i^U})) \\ &\approx \sum_i \sum_d q_\phi(\boldsymbol{z}_{i,t}^U|_d) \left[\log q_\phi(\boldsymbol{z}_{i,t}^U|_d) - \log p(\boldsymbol{z}_{i,T}^U|_d) \right] \end{aligned} \quad (7)$$

Here, since $p(\boldsymbol{b}_i^U)$ is approximately surrogated by an *approximate mixture multivariate normal distribution* in Eq.(6), the KL term is also be approximated in Eq.(7).

Next, we infer the reconstruction loss corresponding to the likelihood $\mathcal{L}_{recon} = -p(r_{i,j}|\mathbf{b_i^U}, \mathbf{b_j^V})$ in Eq.(2). In this paper, we assume that the observed rating data obey the Poission distribution, such that:

$$\mathcal{L}_{recon} = -\frac{1}{r_{i,j}!} \exp\left(r_{i,j} \log\left(s_B(b_i^U, b_j^V)\right) - s_B(b_i^U, b_j^V)\right). \tag{8}$$

where $s_B(b_i^U, b_j^V) = \frac{(\mathbf{b_i^U})^T \mathbf{b_j^V} + K}{2K}$ Considering the equivalence of inner product and Hamming distance, hash representations can be optimized directly by optimizing Eq.(8).

Finally, fusing both user side and item side information, the loss function for the optimal hash representations can be defined as:

$$\mathcal{L}_{rating} = \mathcal{L}_{recon} + \mathcal{L}_{Align}^U + \mathcal{L}_{KL}^U + \mathcal{L}_{Align}^V + \mathcal{L}_{KL}^V \tag{9}$$

It is worth noting that, in Eq.(5), $\boldsymbol{b}_i^U$ ($\boldsymbol{b}_j^V$) is achieved by $\mathrm{sign}(\cdot)$, a non-smooth function that makes the gradient of all inputs to zero during backward propagation. Thus, we adopt an identify function $f(\cdot)$ to surrogate $\mathrm{sign}(\cdot)$, such that $f(\cdot)$ achieves unit gradient in the backward pass.

3.3 Cluster Consistency Preserving Component

To further enhance the quality of both real-valued representations and hash representations, we push them to reach structural consensus. In other words, neighbors generated by real-valued representations and hash representations are expected to be same, such that semantic cluster structure [Liu *et al.*, 2021] can be preserved.

Neighbor Aggregation. In this part, since user side and item side can work in the same way, thus, for concise illustration, we employ user side as a specific example for illustration. Firstly, we aggregate neighbors into different clusters respectively with real-valued representations and hash representations. Due to the characteristics of data independence and time-efficiency, Locality Sensitivity Hashing (LSH) has been proven a powerful approach for approximate nearest neighbor (ANN) search [Qi *et al.*, 2021b]. Thus, we resort to LSH for neighbor aggregation. Concretely, since the intrinsic continuity of real-valued representations and the discreteness of hash representations, the hash functions of the two representations are respectively for Euclidean distance and Hamming distance. Formally, for the real-valued side, we recruit hash functions as:

$$h_i(\boldsymbol{z}_{\boldsymbol{i,0}}^{\boldsymbol{U}}) = \lfloor \frac{\boldsymbol{a} \cdot \boldsymbol{z}_{\boldsymbol{i,0}}^{\boldsymbol{U}} + c}{w} \rfloor \tag{10}$$

where $\boldsymbol{a} \sim \mathcal{N}(0, I)$ is a K-dimentional random vector, $c \sim \mathcal{U}(0, w)$ is a random real value, and $w \in \mathbb{R}^+$ is a hyper-parameter. Since LSH is a probability-based approach, we perform the above hash process L rounds to ensure credibility. Then, the final hash value can be calculated by:

$$H_i(\boldsymbol{z}_{\boldsymbol{i,0}}^{\boldsymbol{U}}) = \sum_{l=1}^{L} B^l h_i^{(l)}(\boldsymbol{z}_{\boldsymbol{i,0}}^{\boldsymbol{U}}) \tag{11}$$

where B is a constant, and $h_i^{(l)}$ is the hash function employed in round l with independently sampled random variables $\boldsymbol{a}$ and c. Then, users with the same hash values will be projected into an identical bucket. We regard neighbors in the same bucket as a cluster. From the obtained cluster structure, we can obtain the similarity matrix $A \in \{1, 0\}^{N_U \times N_U}$ corresponding to real-valued latent representations as:

$$A_{i_1,i_2} = \begin{cases} 1, & H_{i_1}(\boldsymbol{z}_{\boldsymbol{i_1,0}}^{\boldsymbol{U}}) = H_{i_2}(\boldsymbol{z}_{\boldsymbol{i_2,0}}^{\boldsymbol{U}}) \\ 0, & H_{i_1}(\boldsymbol{z}_{\boldsymbol{i_1,0}}^{\boldsymbol{U}}) \neq H_{i_2}(\boldsymbol{z}_{\boldsymbol{i_2,0}}^{\boldsymbol{U}}) \end{cases} \tag{12}$$

where 1 indicates "similar", 0 indicates not. Likewise, for hash representations, we perform the hash functions for Hamming distance L rounds and calculate the final hash values by Eq.(13).

$$h_i^{'(l)}(\boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{U}}) = \boldsymbol{b}_{\boldsymbol{i}}^{(d)}, \quad H_i'(\boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{U}}) = \textstyle\sum_{l=1}^{L} 2^{L-1} h_i^{'(l)} \tag{13}$$

where $h_i^{'(l)}$ denotes hash functions, $H_i'(\boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{U}})$ denotes final hash values, $\boldsymbol{b}_{\boldsymbol{i}}^{(d)}$ indicates the binary value of the d-th dimension in D-dimensional hash representation $\boldsymbol{b}_{\boldsymbol{i}}$. Users (items) with the same hash values are projected into an identical bucket and regarded as a cluster. With the obtained cluster structure, the similarity matrix $A' \in \{1, 0\}^{N_U \times N_U}$ corresponding to hash representations can also be obtained.

Consistency Learning. Till now, joining both user side and item side, the optimization objective of cluster consistency can be defined by:

$$\begin{aligned} \mathcal{L}_{cons} = &\sum_{i_1=1}^{N_U} \sum_{i_2=1}^{N_U} \|\boldsymbol{z}_{\boldsymbol{i_1}}^{\boldsymbol{U}} - \boldsymbol{z}_{\boldsymbol{i_2}}^{\boldsymbol{U}}\|_2 A'_{i_1,i_2} + \sum_{i_1=1}^{N_U} \sum_{i_2=1}^{N_U} (\boldsymbol{b}_{\boldsymbol{i_1}}^{\boldsymbol{U}})^{\boldsymbol{T}} \boldsymbol{b}_{\boldsymbol{i_2}}^{\boldsymbol{U}} A_{i_1,i_2} \\ &+ \sum_{j_1=1}^{N_V} \sum_{j_2=1}^{N_V} \|\boldsymbol{z}_{\boldsymbol{j_1}}^{\boldsymbol{V}} - \boldsymbol{z}_{\boldsymbol{j_2}}^{\boldsymbol{V}}\|_2 A'_{j_1,j_2} + \sum_{j_1=1}^{N_V} \sum_{j_2=1}^{N_V} (\boldsymbol{b}_{\boldsymbol{j_1}}^{\boldsymbol{V}})^{\boldsymbol{T}} \boldsymbol{b}_{\boldsymbol{j_2}}^{\boldsymbol{V}} A_{j_1,j_2} \end{aligned} \tag{14}$$

where, for the user side, the first term indicates that similar users' hash representations require smaller Euclidean distances for users' real-valued representations, while the second term indicates that similar users' real-valued representations require smaller Hamming distances for users' hash representations, which is likewise for the item side.

3.4 Combined Loss Function

Next, we put the two components together and get a fused loss function of HCFRec defined as:

$$\mathcal{L} = \mathcal{L}_{rating} + \lambda \mathcal{L}_{cons} \tag{15}$$

where λ is a hyper-parameter that balances the loss of two components, $\mathcal{L}_{rating}$ is the loss of the first component obtained in Eq.(9), and $\mathcal{L}_{cons}$ is the loss of the second component obtained in Eq.(14).

4 Experiments and Evaluation

We conduct extensive experiments to answer the following questions: **Q1:** How does HCFRec compare with the state-of-art approaches in terms of recommendation accuracy (see subsection 4.3)? **Q2:** How does Hamming distance outperform real-valued inner product in terms of computational efficiency and storage costs (see subsection 4.4)? **Q3:** How does HCFRec perform on parameter sensitivity (see subsection 4.5)?

Datasets		#users	#itmes	#ratings	#sparsity
MovieLens	ML-100K	911	927	47,056	94.428%
	ML-1M	5,954	3,011	496,573	97.230%
	ML-10M	67,976	8,882	4,972,679	99.176%
Amazon	Clothing	6,671	20,755	39,819	99.971%
	Office	838	2,894	11,385	99.531%
	Toys	2,634	10,059	26,293	99.901%

Table 1: Datasets descriptions.

4.1 Data Preparation

We adopt two well-known datasets, MovieLens[1] and Amazon[2] for experimental evaluations: (1) **MovieLens** collects user ratings for movies, ranging from 1 (worst) to 5 (best). We evaluate recommendation performance in terms of different data scale, i.e., *ML-100K*, *ML-1M*, *ML-10M*. (2) **Amazon** [He and McAuley, 2016] covers ratings (with the range of 1 to 5) for up to 24 product categories. We evaluate recommendation performance on its 3 product categories, i.e., *Clothing, Shoes and Jewelry*, *Office Products*, and *Toys and Games*.

We preprocess the data following [Hansen *et al.*, 2020] and [Lian *et al.*, 2017] to filter users and items with less than 20 ratings. Moreover, only the last rating is reserved if a user has rated an item multiple times. We sort the ratings in ascending order according to the feedback time, and divide all datasets into training set, validation set, and test set according to 5:2:3. We summarize the dataset information with Table 1.

4.2 Experimental Settings

Comparison Methods. Five baselines classified into two groups and two versions of our proposed approaches, i.e., HCFRec/no.C and HCFRec, are compared in our experiments, where HCFRec/no.C is the version of HCFRec without cluster consistency preserving component.

(1) *Classical real-valued CF approaches*: **BPR**: A classical recommendation framework maximizing posterior estimator to create a personalized ranking list for a group of items [Rendle *et al.*, 2009]. **BiVAE**: A state-of-the-art bilateral VAE that learns user real-valued representations and item real-valued representations for CF [Truong *et al.*, 2021][3].

(2) *Hash-CF approaches*: **DCF**: An explicit feedback-based Hash-CF method that learns balanced and uncorrelated hash codes for recommendation task [Zhang *et al.*, 2016][4]. **DPR**: A Hash-CF method tha learns hash codes based on personalized ranking objective instead of rating prediction objective in DCF [Zhang *et al.*, 2017a][5]. **BiVAEB**: A version directly binarizing the real-valued representations in BiVAE for recommendation task, which is a baseline indicating the quantization loss from real values to binary representations.

Evaluation Metrics. We focus on the item positions in recommendation lists to evaluate the accuracy of the above competitive methods. In concrete, we adopt the following two metrics: mean Average Precision (mAP) and normalized Discounted Cumulatiive Gain (nDCG).

Parameter Setting. We conduct our experiments on an NVIDIA RTX 3090 GPU by PyTorch. We adopt Adam optimizer with learning rate 0.015 for training. Moreover, the mini-batch SGD with the fixed batch size 128 is employed for optimization. As the introduced hyper-parameters in our model, we set γ in Eq. (6) as 0.015 and $\lambda = 0.3$ in Eq.(15). Following [Zheng *et al.*, 2020], we set $\omega = 8$ in Eq.(10) and $B = 4$ in Eq(11). Moreover, we set $L = 1$ in both Eq (11) and Eq (13), because multiple rounds of training have replaced L to make the obtained hash values confident.

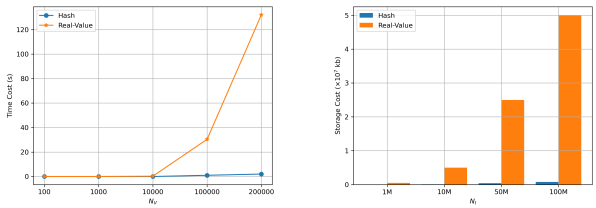
4.3 Performance Comparison (for Q1)

We report the recommendation accuracy in terms of nDCG and mAP respectively in Table 2 and Table 3. Moreover, we evaluate the performance on all methods with both 16-dimensional and 64-bit representations. To illustrate the performance intuitively, we boldly mark the best performance among all Hash-CF methods in each column.

As Table 2 and Table 3 show, HCFRec achieves superior performance over other Hash-CF baselines with improvements of at least 7.18% in terms of nDCG and at leat 20.75% in terms of mAP. Moreover, on Amazon-Clothing and Amazon-Toys datasets (two extremely sparse settings), HCFRec also achieves higher robustness of recommendations over other Hash-CF baselines, which demonstrates a better generalization ability of VAE framework. Furthermore, considering the performance gap between HCFRec/no.C and HCFRec, we find that semantic information takes effect for more informative hash codes.

In addition to Hash-CF approaches, the state-of-the-art real-valued CF (BPR and BiVAE) are also employed for experimental evaluation. As expected, the two real-valued CF methods perform better on all datasets since they leverage more informative representations than Hash-CF methods. However, we can observe that the performance gap becomes smaller when dimension increases. This is because even in low-dimensional scenarios, real-valued CF can still collect rich information for recommendations. In this situation, higher dimensions often bring a limited performance improvement. As a contrast, more dimensions of hash code often result in a higher ability of information representation for better recommendation performances.

4.4 Efficiency and Storage (for Q2)



(a) computational efficiency (b) storage cost

Figure 2: Comparisons of real-valued representations and hash representations in terms of efficiency and storage.

Recall that one of the major motivations of our proposal is to recruit hash representation for more efficient computation

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

[2]http://jmcauley.ucsd.edu/data/amazon/

[3]https://github.com/PreferredAI/bi-vae

[4]https://github.com/hanwangzhang/Discrete-Collaborative-Filtering

[5]https://github.com/yixianqianzy/dpr

	MovieLens-100K						MovieLens-1M						MovieLens-10M					
	16dim			64dim			16dim			64dim			16dim			64dim		
nDCG	@2	@6	@10	@2	@6	@10	@2	@6	@10	@2	@6	@10	@2	@6	@10	@2	@6	@10
BPR	.4840	.4644	.4498	.4815	.4468	.4231	.5194	.4944	.4765	.4607	.4438	.4290	.5607	.5241	.5002	.5619	.5285	.5069
BiVAE	.6599	.6031	.5692	.6415	.5919	.5580	.5953	.5611	.5362	.6008	.5609	.5364	.5665	.5301	.5063	.5542	.5118	.4857
DCF	.3031	.2968	.2854	.3279	.3043	.2966	.2436	.2393	.2342	.3743	.3559	.3487	.1297	.1364	.1183	.1836	.1792	.1723
DPR	.3577	.3423	.3389	.3821	.3702	.3572	.2607	.2526	.2460	.3922	.3687	.3614	.1305	.1402	.1394	.2157	.2248	.2186
BiVAEB	.2254	.2157	.2059	.1656	.1557	.1484	.1821	.1713	.1638	.0958	.0907	.0870	.1179	.1083	.1019	.0986	.0908	.0861
HCFRec/no.C	.4039	.3945	.3815	.4429	.4072	.3849	.2708	.2547	.2503	.4169	.3892	.3725	.1334	.1448	.1575	.2371	.2458	.2447
HCFRec	**.4599**	**.4155**	**.3997**	**.4709**	**.4410**	**.4158**	**.3638**	**.3432**	**.3293**	**.4425**	**.4099**	**.3895**	**.1814**	**.1872**	**.1943**	**.2775**	**.2698**	**.2539**
	Amazon-Clothing						Amazon-Office						Amazon-Toys					
	16dim			64dim			16dim			64dim			16dim			64dim		
nDCG	@2	@6	@10	@2	@6	@10	@2	@6	@10	@2	@6	@10	@2	@6	@10	@2	@6	@10
BPR	.0047	.0041	.0044	.0056	.0054	.0056	.0512	.0504	.0485	.0818	.0761	.0738	.0154	.0143	.0136	.0275	.0260	.0256
BiVAE	.0073	.0067	.0071	.0089	.0076	.0080	.0973	.0867	.0816	.0803	.0792	.0729	.0390	.0352	.0329	.0364	.0326	.0312
DCF	.0015	.0019	.0018	.0019	.0021	.0019	.0256	.0227	.0210	.0496	.0421	.0398	.0095	.0089	.0087	.0187	.0180	.0179
DPR	.0016	.0021	.0022	.0026	.0025	.0025	.0267	.0240	.0239	.0573	.0564	.0552	.0103	.0096	.0093	.0213	.0209	.0199
BiVAEB	.0006	.0005	.0006	.0002	.0004	.0005	.0132	.0143	.0130	.0220	.0207	.0194	.0075	.0063	.0070	.0047	.0045	.0047
HCFRec/no.C	.0022	.0026	.0026	.0030	.0027	.0030	.0270	.0332	.0309	.0693	.0620	.0591	.0128	.0108	.0098	.0233	.0213	.0206
HCFRec	**.0037**	**.0029**	**.0034**	**.0041**	**.0035**	**.0037**	**.0478**	**.0474**	**.0478**	**.0779**	**.0748**	**.0723**	**.0134**	**.0131**	**.0135**	**.0242**	**.0224**	**.0221**

Table 2: Experimental results on datasets.

	MovieLens-100K		MovieLens-1M		MovieLens-10M		Amazon-Clothing		Amazon-Office		Amazon-Toys	
mAP@10	16dim	64dim	16dim	64dim	16dim	64dim	16dim	64dim	16dim	64dim	16dim	64dim
BPR	.0801	.0772	.0556	.0520	.0865	.0917	.0016	.0021	.0112	.0189	.0032	.0071
BiVAE	.1159	.1136	.0699	.0713	.0909	.0832	.0025	.0030	.0213	.0182	.0093	.0096
DCF	.0427	.0431	.0135	.0289	.0108	.0193	.0005	.0007	.0054	.0125	.0013	.0049
DPR	.0496	.0501	.0143	.0316	.0126	.0208	.0006	.0008	.0059	.0133	.0015	.0053
BiVAEB	.0227	.0139	.0100	.0055	.0060	.0059	.0002	.0001	.0026	.0043	.0011	.0013
HCFRec/no.C	.0581	.0585	.0176	.0369	.0131	.0239	.0009	.0010	.0062	.0148	.0024	.0061
HCFRec	**.0642**	**.0670**	**.0283**	**.0394**	**.0184**	**.0293**	**.0013**	**.0014**	**.0106**	**.0175**	**.0026**	**.0064**

Table 3: Experimental results on datasets.

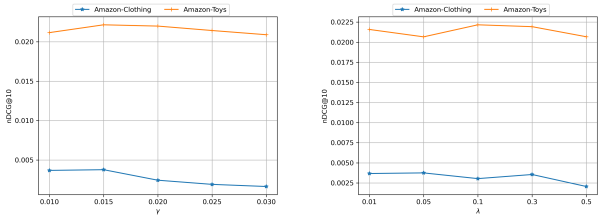
and lighter storage. In this subsection, we respectively investigate computational efficiency and storage cost as follows.

Computational Efficiency. We fix the user size to 100,000 and change the item size N_V from 100 to 200,000. The 64-dimensional representations of real-value and hash codes are randomly generated. The evaluation results are illustrated in Figure 2(a): computational time of real-valued representations grows exponentially as item scale increases; as a contrast, the time cost of hash representations achieves a speedup factor of 40-50. This observation demonstrates the efficiency benefit of Hash-CF for large-scale recommendation.

Storage Costs. We compare the storage costs in Figure 2(b) with item sizes from 1 million to 1 billion. As expected, storing real-valued representations consumes more space than hash codes. This is because one dimension of real-valued representation often needs 64 bits, while one dimension of hash code needs only 1 bit. This fact highlights the space thrift of Hash-CF for large-scale recommendation.

4.5 Parameter Sensitivity (for Q3)

We investigate the recommendation performances influenced by hyper-parameters γ and λ in Figure 3, during which the 64-dimensional hash representations are adopted and the top-10 items are returned for evaluation nDCG performance on the two sparsest datasets, i.e., Amazon-Clothing and Amazon-Toys. With fixed $\lambda = 0.3$, we vary γ from 0.010 to 0.030. Figure 3(a) shows that our model achieves the best performance when $\gamma = 0.015$. This is because a continuous distribution extremely close to a discrete distribution (with γ smaller than 0.015) is too complex to learn. As a contrast, the much smoother distribution (with γ larger than 0.015) leads to more quantization errors in the latter quantization step, so as to dampen the recommendation performance. Furthermore, we fix $\gamma = 0.015$ and evaluate nDCG@10 with λ verying from 0.01 to 0.5. Figure 3(b) shows that the performances are not sensitive to λ. Therefore, we set $\gamma = 0.015$ and $\lambda = 0.3$.



(a) sensitivity of γ (b) sensitivity of λ

Figure 3: nDCG@10 w.r.t. different γ and λ.

5 Conclusion

In the big data environment, Hash-CF has been proven a promising technique to accelerate recommendation efficiency by learning an optimal hash representation. However, traditional Hash-CF often falls short in the optimization on discrete hash representations and the preservation of semantic information. To tackle these issues, we introduce normalized flow to learn the optimal hash code and deploy a cluster consistency preserving mechanism to preserve the semantic structure in representations. Extensive experiments conducted on six real-world datasets reveal the superiority of our proposal in terms of accuracy and efficiency.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (No.2018YFB1403001), the National Natural Science Foundation of China (No.62172362 and No.72192823) and Leading Expert of "Ten Thousands Talent Program" of Zhejiang Province (No.2021R52001).

References

[Chen *et al.*, 2018] Chaochao Chen, Ziqi Liu, Peilin Zhao, Longfei Li, Jun Zhou, and Xiaolong Li. Distributed collaborative hashing and its applications in ant financial. In *ACM SIGKDD*, 2018.

[Chen *et al.*, 2022] Chaochao Chen, Huiwen Wu, Jiajie Su, Lingjuan Lyu, Xiaolin Zheng, and Li Wang. Differential private knowledge transfer for privacy-preserving cross-domain recommendation. *arXiv preprint arXiv:2202.04893*, 2022.

[Hansen *et al.*, 2020] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. Content-aware neural hashing for cold-start recommendation. In *ACM SIGIR*, 2020.

[He and McAuley, 2016] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, 2016.

[Jordan *et al.*, 1999] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[Karamanolakis *et al.*, 2018] Giannis Karamanolakis, Kevin Raji Cherian, Ananth Ravi Narayan, Jie Yuan, Da Tang, and Tony Jebara. Item recommendation with variational autoencoders and heterogeneous priors. In *ACM DLRS*. Association for Computing Machinery, 2018.

[Karatzoglou *et al.*, 2010] Alexandros Karatzoglou, Alexander Smola, and Markus Weimer. Collaborative filtering on a budget. *Journal of Machine Learning Research-Proceedings Track*, 9:389–396, 2010.

[Kingma and Welling, 2014] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, 2014.

[Lee *et al.*, 2017] Wonsung Lee, Kyungwoo Song, and Il-Chul Moon. Augmented variational autoencoders for collaborative filtering with auxiliary information. In *ACM CIKM*, 2017.

[Lian *et al.*, 2017] Defu Lian, R. Liu, Yong Ge, Kai Zheng, Xing Xie, and Longbing Cao. Discrete content-aware matrix factorization. In *ACM SIGKDD*, 2017.

[Liang *et al.*, 2018] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *WWW*, 2018.

[Liu *et al.*, 2019] Chenghao Liu, Tao Lu, Xin Wang, Zhiyong Cheng, Jianling Sun, and Steven CH Hoi. Compositional coding for collaborative filtering. In *ACM SIGIR*, 2019.

[Liu *et al.*, 2021] Weiming Liu, Jiajie Su, Chaochao Chen, and Xiaolin Zheng. Leveraging distribution alignment via stein path for cross-domain cold-start recommendation. In *NeurIPS*, volume 34, 2021.

[Qi *et al.*, 2021a] Lianyong Qi, Chunhua Hu, Xuyun Zhang, Mohammad R. Khosravi, Suraj Sharma, Shaoning Pang, and Tian Wang. Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment. *IEEE Transactions on Industrial Informatics*, 17(6):4159–4167, 2021.

[Qi *et al.*, 2021b] Lianyong Qi, Xiaokang Wang, Xiaolong Xu, Wanchun Dou, and Shancang Li. Privacy-aware cross-platform service recommendation based on enhanced locality-sensitive hashing. *IEEE Transactions on Network Science and Engineering*, 8(2):1145–1153, 2021.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *ACM UAI*, 2009.

[Rezende and Mohamed, 2015] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[Shan *et al.*, 2018] Ying Shan, Jian jiao, Jie Zhu, and JC Mao. Recurrent binary embedding for gpu-enabled exhaustive retrieval from billion-scale semantic vectors. In *ACM SIGKDD*, 2018.

[Truong *et al.*, 2021] Quoc-Tuan Truong, Aghiles Salah, and Hady W Lauw. Bilateral variational autoencoder for collaborative filtering. In *WSDM*, 2021.

[Wang *et al.*, 2018] Jingdong Wang, Ting Zhang, jingkuan song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, 2018.

[Zhang *et al.*, 2014] Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. Preference preserving hashing for efficient recommendation. In *ACM SIGIR*, 2014.

[Zhang *et al.*, 2016] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. Discrete collaborative filtering. In *ACM SIGIR*, 2016.

[Zhang *et al.*, 2017a] Yan Zhang, Defu Lian, and Guowu Yang. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *IEEE AAAI*, 2017.

[Zhang *et al.*, 2017b] Yan Zhang, Guowu Yang, Lin Hu, Hong Wen, and Jinsong Wu. Dot-product based preference preserved hashing for fast collaborative filtering. In *IEEE ICC*, 2017.

[Zheng *et al.*, 2020] Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *ArXiv*, abs/2011.09315, 2020.

[Zhu *et al.*, 2021] Feng Zhu, Yan Wang, Jun Zhou, Chaochao Chen, Longfei Li, and Guanfeng Liu. A unified framework for cross-domain and cross-system recommendations. *CoRR*, 2021.