Collaborative Hashing

Xiangl Liu[†] Junfeng He[‡] Cheng Deng[♮] Bo Lang[†]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[‡]Facebook, 1601 Willow Rd, Menlo Park, CA, USA
[♮]Xidian University, Xi'an, China
{xlliu, langbo}@nlsde.buaa.edu.cn jfh@fb.com chdeng.xd@gmail.com

Abstract

Hashing technique has become a promising approach for fast similarity search. Most of existing hashing research pursue the binary codes for the same type of entities by preserving their similarities. In practice, there are many scenarios involving nearest neighbor search on the data given in matrix form, where two different types of, yet naturally associated entities respectively correspond to its two dimensions or views. To fully explore the duality between the two views, we propose a collaborative hashing scheme for the data in matrix form to enable fast search in various applications such as image search using bag of words and recommendation using user-item ratings. By simultaneously preserving both the entity similarities in each view and the interrelationship between views, our collaborative hashing effectively learns the compact binary codes and the explicit hash functions for out-of-sample extension in an alternating optimization way. Extensive evaluations are conducted on three well-known datasets for search inside a single view and search across different views, demonstrating that our proposed method outperforms state-of-the-art baselines, with significant accuracy gains ranging from 7.67% to 45.87% relatively.

1. Introduction

Recently hashing technique has attracted great attentions in fast similarity search [3, 5, 12–15, 18]. Based on the concept of locality sensitivity [7], it represents data points using the binary codes that preserve the original similarities among data. The retrieval of similar data points can then be completed in a sublinear or even constant time, by Hamming distance ranking based on fast binary operation or hash table lookup within a certain Hamming distance. Moreover, the binary compression of original data can largely reduce the storage consumption.

Existing hashing approaches usually find the compact binary codes by exploiting the data correlations among the data entities (*e.g.*, the cosine similarities between feature vectors). In practice, there are many scenarios involving nearest neighbor search on the data matrix with two dimensions corresponding to two different coupled views or entities. For instance, the classic textual retrieval usually works based on the term-document matrix with each element representing the correlations between two views: words and documents. Recently, such bag-of-words (BoW) model has also been widely used in computer vision and multimedia retrieval, which mainly captures the correlations between local features (even the dictionaries using sparse coding) and visual objects [16, 17]. Besides the search applications, the rapidly developed recommendation studies based on collaborative filtering, analyzing relationships between users and interdependencies among items to identify new user-item associations, also impose strong requirements on the nearest neighbor search with a collected rating matrix between users and items [2, 10, 20].

In the literature, rare works have been reported to regard the hashing problem with the matrix-form data, considering the duality between different types of entities or views. It is difficult to directly apply the traditional hashing methods to simultaneously hashing two types of entities and meanwhile preserving their correlations. Recently, [19] and [20] attempted to address the problem by concentrating on the duality between views under different scenarios. [19] viewed both documents and terms as the same type of entities in a bipartite graph, and pursued the binary codes for them following the idea of spectral hashing [18]. To handle the unobserved/missing ratings in collaborative filtering, [20] directly learned the binary codes that can recover the observed item preferences of all users. Both methods mainly explore the correlations between two views in order to preserve the occurrences or preferences in the training matrix. However, they neglected the fact that the entities' neighbor structures inside each view play rather important roles in compact binary codes pursuit in nearest neighbor search [4, 6] or collective patterns discovery in

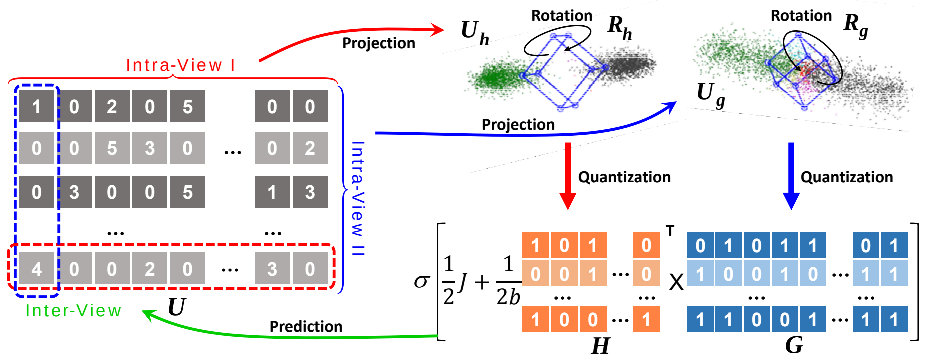


Figure 1. The proposed collaborative hashing for nearest neighbor search over data in matrix form.

collaborative filtering [10, 20].

In this paper we propose a collaborative hashing (CH) scheme for nearest neighbor search with data in matrix form. By simultaneously preserving both the inner similarities in each view (intra-view) and the interrelationships across views (inter-view), it learns the compact hash codes and the hash functions for entities in two views. As Figure 1 demonstrates, the proposed collaborative scheme fundamentally works as follows: On the one hand, fusing the semantic correlations between the coupled views will enhance the discriminative power of the features in each view, and subsequently helps obtain more compact and meaningful hash codes for each view. On the other hand, by retaining the duality between two correlated views based on the collective patterns discovered through hash grouping in each view, the collaborative hashing embeds different types of entities into a joint semantic Hamming space, which enables fast and accurate search across views.

Therefore, the proposed collaborative hashing can serve as a unified framework for applications including: (1) **search inside a single view**: the most typical example is the visual search using local descriptors [16, 17]; (2) **search across different views**: Recommendation using user-item ratings falls in this direction [2, 10, 20]. In many successful recommendation algorithms, matrix factorization serves as a core technique for collaborative filtering [10]. It turns the item suggestions into the search based on their similarities in a low dimensional Euclidean space. Instead of the Euclidean space, our collaborative hashing aims to find binary codes in Hamming space, guaranteeing efficient storage and fast similarity search across users and items.

We essentially summarize our contributions as follows:

1. We propose a collaborative hashing scheme general for nearest neighbor search with data in matrix form. By simultaneously considering both the intra-view (in each view) and the inter-view (across views) relationships in one framework, it enables fast similarity search widely involved in many applications such as image search using bag of words and recommendation using user-item ratings.

2. We formulate it as an optimization problem with a loss function, which incorporates the binary quantization loss for the neighbor relations in each view, and the divergence between the training data and the predictions based on the binary codes for the interrelationships across views.

3. We adopt the projection-based linear hash functions, and present an alternating optimization way to effectively learn both the binary codes and the explicit hash functions for out-of-sample extension.

Our extensive empirical study on several large-scale benchmarks highlights the benefits of our method for visual search and item recommendation, with significant performance gains over state-of-the-art hashing methods.

2. Collaborative Hashing

2.1. Notation

Suppose we have the data matrix $U = (u_{ij}) \in \mathbb{R}^{n\times m}$, consisting of n entities (images, users, etc.) of type I in one dimension and m entities (visual words, items, etc.) of type II in another dimension. Therefore, the matrix actually characterizes the associations between the two types of entities in terms of observations like word counts or user-item ratings: Each entry u_{ij} in the matrix indicates the presence of entity j of type II in entity i of type I, and its value reflects the strength of their association.

Existing hashing algorithms either independently treat the two views of the data matrix [2,4], or mainly concentrate on the correlation information between the two views [19, 20]. In our collaborative hashing, we will simultaneously learn the discriminative hash functions for both views, which not only preserve the locality in each view, but also capture the correlations between views.

Specifically, the problem of collaborative hashing can be formally defined as respectively learning b hash functions $h_i(\cdot), i = 1,\ldots,b$ and $g_j(\cdot), j = 1,\ldots,b$ for different types of entities, and generating the corresponding compact binary codes $H = [\mathbf{h}_1 \ \ldots \ \mathbf{h}_n] \in \{-1,+1\}^{b\times n}$ and $G = [\mathbf{g}_1 \ \ldots \ \mathbf{g}_m] \in \{-1,+1\}^{b\times m}$ that well preserve the semantic relationships conveyed by the matrix U. For simplicity, we apply the linear-form hash functions

$h_i(\mathbf{x}) = \text{sign}(\mathbf{w}_i^T\mathbf{x})$ and $g_i(\mathbf{x}) = \text{sign}(\mathbf{v}_i^T\mathbf{x})$ where $W = [\mathbf{w}_1 \ \dots \ \mathbf{w}_b]$ and $V = [\mathbf{v}_1 \ \dots \ \mathbf{v}_b]$ are the projection parameters for both types of hash functions.

2.2. Formulation

Our basic idea is trying to exploit the intrinsic relations in the matrix data: both the neighbor structures in each view and the semantic correlations between views. Correspondingly, two types of losses will be employed to guide the hash function learning in our formulation: the binary quantization loss of hash functions and the correlation prediction loss using the binary codes.

Firstly, we respectively treat each row and column of U as the features of the corresponding entities of type I and type II, and hash them into binary codes. Note that the hashing process, grouping similar entities for the collective patterns discovery, serves as an important characteristic of the collaborative scheme. Since the binary quantization in hashing behaves like the clustering algorithms [4, 6, 15], minimizing such quantization loss faithfully preserves the similarity structure of the data in each view:

$$E_{\text{quan}} = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{h}_i - R_h\tilde{\mathbf{u}}_i\|^2 + \frac{1}{m}\sum_{j=1}^{m}\|\mathbf{g}_j - R_g\hat{\mathbf{u}}_j\|^2. \quad (1)$$

Here $U_h = [\tilde{\mathbf{u}}_1 \ \dots \ \tilde{\mathbf{u}}_n] \in \mathbb{R}^{b\times n}$ and $U_g = [\hat{\mathbf{u}}_1 \ \dots \ \hat{\mathbf{u}}_m] \in \mathbb{R}^{b\times m}$ are the compressed feature representations for the involved two types of entities (see the initialization details in Section 2.3.4). $R_h \in \mathbb{R}^{b\times b}$ and $R_g \in \mathbb{R}^{b\times b}$ are orthogonal rotation matrices for better quantization:

$$R_h^T R_h = I \text{ and } R_g^T R_g = I. \quad (2)$$

Besides the quantization error along each view, we should also explore the duality between the two views. Intuitively, the learned binary codes for two views should also preserve their correlations (the word counts in BoW features, the ratings in recommendation, etc.) in the Hamming space, rather than the latent Euclidean space in traditional collaborative filtering. Consequently, the correlation between entity i of type I and entity j of type II can be predicted efficiently in Hamming space [12, 20]:

$$c(\mathbf{h}_i, \mathbf{g}_j) = \frac{1}{2} + \frac{1}{2b}\mathbf{h}_i^T\mathbf{g}_j. \quad (3)$$

with the prediction scaling variable σ, the correlation prediction error on all the training samples can be given by:

$$E_{\text{rate}} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\sigma c(\mathbf{h}_i, \mathbf{g}_j) - u_{ij}\right)^2. \quad (4)$$

Here, we first consider the basic case that all elements in U are observed. A well-known example is the bag-of-words feature, of which each element is regarded to be observed occurrence. For the sparse observation, we will give a simple, yet efficient solution in Section 2.5.2.

Putting the quantization error and prediction error together, we can minimize the following objective function

$$E = E_{\text{quan}} + \lambda E_{\text{rate}}, \quad (5)$$

where λ is a fixed weight, balancing the importance of similarities in a single view and across views. By rewriting the objective in matrix form with an $n \times m$ matrix of ones J, we can get

$$\begin{aligned} \min_{H,G,R_h,R_g,\sigma} \quad & \frac{1}{n}\|H - R_hU_h\|_F^2 + \frac{1}{m}\|G - R_gU_g\|_F^2 \\ & + \frac{\lambda}{nm}\left\|\sigma\left(\frac{1}{2}J + \frac{1}{2b}H^TG\right) - U\right\|_F^2 \\ \text{s.t.} \quad & H \in \{-1,+1\}^{b\times n},\ G \in \{-1,+1\}^{b\times m} \\ & R_h^TR_h = I,\ R_g^TR_g = I. \end{aligned} \quad (6)$$

To further reduce the redundancies within the binary codes H and G, we introduce the uncorrelated constraints:

$$HH^T = nI \text{ and } GG^T = mI. \quad (7)$$

Note that the above formulation reveals an interesting connection between learning binary codes and factorizing the matrix in collaborative filtering. In particular, both methods approximate the correlations in low-dimensional space: when σ is already known, the last term in (6) is quite similar to the factorization loss in collaborative filtering. But our collaborative hashing here mainly focuses on the Hamming space for efficient storage and similarity search.

Next we will give an alternating optimization solution to (6), which can efficiently learn both the binary codes and the explicit hash functions for out-of-sample extension.

2.3. Optimization

The optimization problem can be solved by alternating among variables. We sequentially describe the updating steps for each variable, assuming the others are fixed.

2.3.1 The Binary Codes

Fixing other variables and using the fact that the binary codes H are constrained to be uncorrelated in (7), we can rewrite the objective function in (6) as

$$\begin{aligned} \max_{H} \quad & \text{trace}\left(H^TD_h\right) \\ \text{s.t.} \quad & HH^T = nI,\ H \in \{-1,+1\}^{b\times n} \end{aligned} \quad (8)$$

where $D_h = \frac{1}{n}R_hU_h + \frac{\lambda\sigma}{2bnm}G\left(U^T - \frac{\sigma}{2}J^T\right)$. By relaxing the discrete constraint on H, the problem can be solved efficiently by the singular value decomposition (SVD): $H = \text{sign}(\tilde{S}_h S_h^T)$ with the SVD $D_h = \tilde{S}_h \Pi S_h^T$.

Likewise, we can update hash codes G by solving

$$\begin{aligned} \max_{G} \quad & \text{trace}\left(G^T D_g\right) \\ \text{s.t.} \quad & GG^T = mI,\ G \in \{-1,+1\}^{b\times m} \end{aligned} \tag{9}$$

with $D_g = \dfrac{1}{m} R_g U_g + \dfrac{\lambda\sigma}{2bnm} H\left(U - \dfrac{\sigma}{2} J\right)$. The near-optimal solution $G = \text{sign}(\tilde{S}_g S_g^T)$, given $D_g = \tilde{S}_g \Lambda S_g^T$.

2.3.2 The Rotation Matrices

With the learned hash codes for both views, we can rewrite the objective function with respect to R_h as

$$\begin{aligned} \min_{R_h} \quad & \|H - R_h U_h\|_F^2 \\ \text{s.t.} \quad & R_h^T R_h = I \end{aligned} \tag{10}$$

The optimal R_h in the above problem can be obtained by solving SVD of matrix HU_h^T: Decomposing HU_h^T as $\tilde{T}_h \Delta T_h^T$, then $R_h = \tilde{T}_h T_h^T$.

For the rotation matrix R_g in another view, we efficiently solve a similar optimization problem as follows:

$$\begin{aligned} \min_{R_g} \quad & \|G - R_g U_g\|_F^2 \\ \text{s.t.} \quad & R_g^T R_g = I \end{aligned} \tag{11}$$

Based on the decomposition of $GU_g^T = \tilde{T}_g \Omega T_g^T$, we obtain the optimal rotation $R_g = \tilde{T}_g T_g^T$.

2.3.3 The Prediction Scalar

The range of the observations in training data varies in different scenarios (*e.g.*, $[0, +\infty)$ in BoW features, and 1-5 in recommendation), while our correlation prediction $c(\cdot,\cdot) \in [0,1]$. Therefore, a scalar variable should be introduced to match the scales between the predictions and the true observations. By fixing other variables, the optimization problem with respect to the scalar σ turns to

$$\min_{\sigma} \quad \|\sigma\left(\frac{1}{2}J + \frac{1}{2b}H^T G\right) - U\|_F^2. \tag{12}$$

With $M = \dfrac{1}{2}J + \dfrac{1}{2b}H^T G$ and the vector representation of the matrix, the problem is equivalent to a least square problem

$$\min_{\sigma} \quad \|\sigma \text{vec}(M) - \text{vec}(U)\|_F^2, \tag{13}$$

whose close-form solution will be

$$\sigma = \frac{\text{vec}(M)^T \text{vec}(U)}{\text{vec}(M)^T \text{vec}(M)}. \tag{14}$$

2.3.4 Initialization

In practice, hashing based on principle component analysis (PCA) can give good initializations of H and G [4]. Specifically, suppose we have the mean vectors μ_h and μ_g of the features U^T and U respectively for the two types of correlated entities, then we can obtain the projection matrices P_h and P_g by taking the top b eigenvectors of the covariance matrix $(U^T - \mu_h \mathbf{1}^T)^T (U^T - \mu_h \mathbf{1}^T)$ and $(U - \mu_g \mathbf{1}^T)^T (U - \mu_g \mathbf{1}^T)$, which preserve the most information in corresponding feature spaces. Based on the projection matrices, we can initialize the compressed feature representations U_h and U_g by

$$U_h = P_h^T (U^T - \mu_h \mathbf{1}^T) \text{ and } U_g = P_g^T (U - \mu_g \mathbf{1}^T). \tag{15}$$

The orthogonal rotation matrices R_h and R_g will not change the variances of U_h and U_g. With the randomly generated R_h and R_g, the binary codes can be initialized:

$$H = \text{sign}(R_h U_h) \text{ and } G = \text{sign}(R_g U_g). \tag{16}$$

2.4. Out-of-Sample Extension

The binary codes for the training entities are learned during the training stage of collaborative hashing. Nevertheless, it is usually required to handle the out-of-sample extension problem in many applications [4, 18]. In our method, we also learn the near-optimal rotation matrices R_h and R_g that balance the binary code assignment to preserve the neighbor relationships globally. Therefore, we can encode the out-of-sample data efficiently using the linear transformations composed of the rotations and the aforementioned projections. This indeed makes our method distinct from the most related work [20] that can only learn binary codes for the training data.

Specifically, for hash functions h_i and g_i, $i = 1, \ldots, b$, the projection parameters are given by $W = P_h R_h^T$ and $V = P_g R_g^T$. Given a new entity $\mathbf{x}$ from any view, we can encode it using the corresponding hash functions

$$h_i(\mathbf{x}) = \text{sign}\left(\mathbf{w}_i^T (\mathbf{x} - \mu_h)\right), \tag{17}$$

$$g_i(\mathbf{x}) = \text{sign}\left(\mathbf{v}_i^T (\mathbf{x} - \mu_g)\right). \tag{18}$$

We list our Collaborative Hashing (CH) in Algorithm 1, which empirically converges fast in less than 20 iterations.

2.5. Applications

As aforementioned, the proposed collaborative hashing serves as a general framework for applications following different search paradigms.

2.5.1 Search Inside a Single View

First, CH can be directly used to address the standard nearest neighbor search problem widely involved in information retrieval and computer vision. The search process

Algorithm 1 Collaborative Hashing (CH).

```
1: Initialize $U_h$, $U_g$ by (15), and $H$, $G$ by (16);
2: repeat
3:     Estimate $\sigma$ according to (14);
4:     Update the binary codes $H$ and $G$ respectively according to (8) and (9);
5:     Update the rotation matrices $R_h$ and $R_g$ respectively according to (10) and (11);
6: until Converge
7: Return the binary codes $H$, $G$, and hash functions $h$ and $f$ according to (17) and (18).
```

is conducted on the same type of entities (*i.e.*, inside a single view). In computer vision, a well-known example is the image search, where given a query image we expect to get the most similar ones from a large database. In these applications, the feature matrix U is usually described based on the bag-of-words model, and each element u_{ij} represents the occurrence of word j in image i.

More recently, [11] studied the cross-modal relationships between words and images, beyond simply identifying the word presence. Similarly, CH can also capture the semantic correlations between the coupled views. Unlike prior studies, we mainly focus on the pursuit of discriminative hash codes for fast similarity search inside each view.

The search scheme here is exactly the same to that of traditional hashing methods using binary codes. With the hash functions given in (17) or (18), for any query the nearest neighbors in the same view can be found efficiently by Hamming distance ranking or hash table lookup.

2.5.2 Search Across Different Views

Another search paradigm supported by our CH is seeking highly correlated entities of different types (*i.e.*, across different views). The collaborative filtering for recommendation is one typical example that suggests the items with highest predicted ratings for specific users. In collaborative filtering, matrix factorization exploits the collective taste patterns from the user-item rating matrix, and improves the search quality by preserving preferences of users over items in a low-dimensional Euclidean space [10]. However, searching over a large database in the continuous space is quite slow. The compact binary codes learned by CH can hopefully help address the problem in a Hamming space.

In recommendation applications, the data matrix U records the user-item preferences: each nonzero elements u_{ij} represents the rating given to the item j by user i. Since there are lots of unobserved/missing user-item ratings in practice, the observation matrix U will be quite sparse. To tackle such case, only the observed data should be considered in the optimization including (8), (9) and (12). We replace the correlation prediction error in (4) by

$$E_{\text{rate}} = \frac{1}{\|A\|_1}\|\sigma\left(\frac{1}{2}J + \frac{1}{2b}H^T G\right) \circ A - U\|_F^2, \qquad (19)$$

where $\circ$ is the Hadamard product operator, and $A = (a_{ij}) \in \{0,1\}^{n \times m}$ is a binary matrix with nonzero entries a_{ij} indicating whenever u_{ij} is observed (usually $\|A\|_1 \ll nm$). Regarding the estimation of σ, instead of (14) we have

$$\sigma = \frac{\text{vec}(M \circ A)^T \text{vec}(U)}{\text{vec}(M \circ A)^T \text{vec}(M \circ A)} \qquad (20)$$

With the hash codes H for users and G for items, to user i we can recommend the item j with largest $c(\mathbf{h}_i, \mathbf{g}_j)$. Therefore, the recommendation problem can also be solved using Hamming distance ranking or hash table lookup.

3. Experiments

In this section we will evaluate the proposed collaborative hashing for two useful scenarios: *search with bag-of-word features* and *recommendation using user-item ratings*, which respectively correspond to the two search paradigms mentioned above. The proposed collaborative hashing method (CH) will be compared with both a number of state-of-the-arts standard hashing methods such as Locality Sensitive Hashing (LSH) [3], Spectral Hashing (SH) [18], Iterative Quantization (ITQ) [4], and Random Maximum Margin Hashing (RMMH) [9], and existing matrix hashing methods including Laplacian Co-Hashing (LCH) [19] and Binary Codes for Collaborative Filtering (BCCF) [20].

LCH simultaneously hashes both terms and documents according to their semantic similarities. Following the idea of spectral hashing, it learns the hash codes by solving an eigenvalue problem for the term-document similarity graph. BCCF was proposed to generalize the existing hashing works to the context of collaborative filtering. It learns binary codes for both users and items by forcing them to accurately preserve the item preferences of users.

To evaluate the hashing performance, we employ two common nearest neighbor search methods following prior hashing research: Hamming distance ranking and hash table lookup. The former ranks all points in the database according to the Hamming distances from the query, while the later constructs a lookup table using the binary codes, and returns points falling within certain Hamming radius (usually 2) from the query codes as the retrieved results.

All experiments are conducted on a workstation with Intel Xeon CPU E5645@2.40GHz and 24GB memory, and the results reported in this paper are averaged over ten independent training/testing data splits. For parameter sensitivity, our experiments indicate that CH is relatively robust with respect to λ in a large range. Thus we roughly set it to 1,000 in all experiments. For all baselines, we fine tuned their parameters for the best performance.

Table 1. MAP (%) of different hashing algorithms using 32 - 128 bits on Holidays dataset.

Holidays	Hash	32 bits	64 bits	128 bits
+15K	LSH	1.63±0.29	3.63±0.34	7.05±0.51
	SH	12.65±0.67	16.36±0.60	21.19±1.32
	RMMH	8.00±0.87	11.28±1.40	16.56±1.64
	ITQ	18.86±0.75	25.42±0.61	30.73±0.72
	LCH	17.96±0.47	25.95±0.72	31.87±0.50
	CH	**20.95**±0.82	**27.54**±0.49	**32.34**±1.30
+25K	LSH	1.01±0.19	1.68±0.24	4.38±0.56
	SH	8.84±1.13	11.76±0.75	15.59±1.05
	RMMH	5.39±1.03	7.59±1.14	11.60±0.40
	ITQ	16.49±0.81	22.92±0.96	28.52±0.71
	LCH	12.67±1.13	20.94±0.82	29.73±0.77
	CH	**18.42**±0.56	**25.61**±0.64	**31.23**±0.51
+100K	LSH	0.56±0.12	1.14±0.13	2.69±0.19
	SH	8.00±0.91	8.03±1.13	9.09±0.57
	RMMH	4.22±0.43	4.98±0.36	7.67±0.93
	ITQ	12.34±0.60	17.21±1.38	22.18±1.11
	LCH	8.67±0.83	15.64±1.27	22.41±0.70
	CH	**12.53**±0.62	**18.53**±0.80	**24.02**±0.23

3.1. Search with Bag of Words

3.1.1 Datasets and Protocols

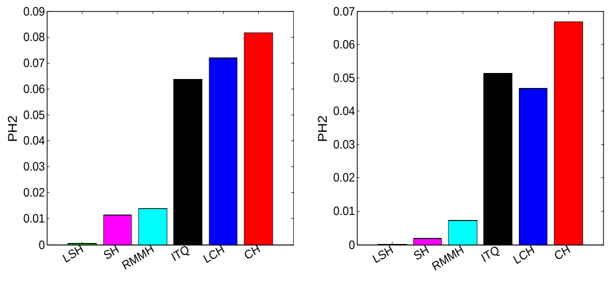
Hashing is widely adopted in visual search based on bag of visual words or high dimensional sparse codes. To evaluate our collaborative hashing in visual search following the paradigm "search inside a single view", we employ the popular Holidays image dataset [8]. It mainly contains 1,491 personal holidays photos of high resolutions: 500 queries representing distinct scenes or objects and 991 corresponding relevant images with various attacks (*e.g.*, rotations, blurring, etc.) to the queries. The dataset covers a very large variety of scene types.

Similar to [1, 8], for large-scale retrieval the Holidays dataset is respectively appended with 15K, 25K and 100K Flickr images (forming Holidays+15K, +25K, and +100K). We represent Holidays+15K images using 10K visual vocabularies, and Holidays+15K and +100K using 20K ones (trained based on SIFT features of an independent dataset Flickr60K [8]). In all experiments, 500 queries in Holidays serve as the testing images, and 10K images are randomly sampled from each dataset as the training sets.

Following prior hashing research [4, 12], we will report both the precision and recall performance of Hamming distance ranking, and the precision within Hamming radius 2 (PH2) when using hash table lookup.

3.1.2 Results and Discussions

We will compare our collaborative hashing to the traditional state-of-the-art hashing methods (LSH, SH, RMMH, and ITQ). As to existing hashing methods for data in matrix form, since BCCF cannot tackle the out-of-sample hashing problem, in our experiments we will only adopt LCH that co-hashes both types of entities in the data matrix.



(a) Holidays+15K (b) Holidays+25K

Figure 3. PH2 using 32 bits on Holidays datasets.

Table 1 presents the mean average precisions (MAP) of different hashing methods on three Holidays datasets. Performances of all methods improve when using longer hash codes, and clearly CH achieves the highest performance in all cases with remarkable superiority (up to 11.08%, 11.80%, and 7.67% performance gains on three datasets over the best competitors).

We also compare precision-recall (P-R) performances using 64 bits on all datasets in Figure 2. LCH explores the duality between images and visual words, and achieves better performance than LSH, RMMH and SH. However, its performance is inferior to that of ITQ in most cases, which indicates that the utilization of inner relationships among images plays an indispensable role in the performance improvement. We have the same conclusion from Table 1 by comparing performances of ITQ and LCH using less than 128 hash bits. Here, collaborative hashing, incorporating the correlations between views in addition to the neighbor structure of each view, boosts the discriminative power of the learned hash codes, and thereby obtains the best performance (the largest areas under the curves) in all cases.

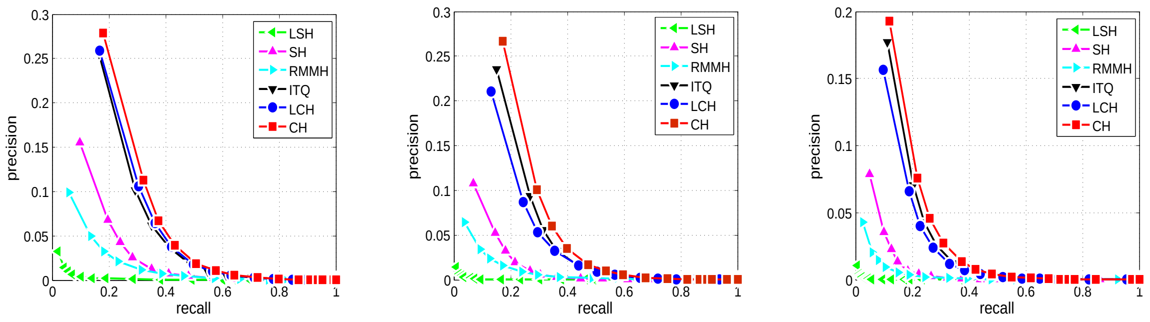
Besides Hamming ranking evaluation, we also conduct hash table lookup on these datasets, and report retrieval precisions within Hamming radius 2 (PH2) in Figure 3. Although in our experiments as the number of distractor images increases from 15K to 100K, performances of all hashing methods drop, but in all cases CH consistently outperforms other methods with large margins.

3.2. Recommend using User-Item Ratings

3.2.1 Datasets and Protocols

Besides the basic search along a single view, an attractive advantage of collaborative hashing is that the learned compact hash codes, well preserving the correlations between the two views, can efficiently and accurately predict the ratings of unseen items in recommendation systems. We adopt two datasets to evaluate the recommendation performance of CH: MovieLens 1M and Netflix, whose statistics are summarized as follows:

- The MovieLens 1M dataset contains 3,900 movies, 6,040 users and about 1 million ratings. In this dataset,



(a) P-R @ 64 bits on Holidays+15K (b) P-R @ 64 bits on Holidays+25K (c) P-R @ 64 bits on Holidays+100K

Figure 2. Performance comparison of different hashing methods using Hamming ranking on Holidays datasets.

about 4% of the user-movie ratings are observed. The ratings are integers ranging from 1 (bad) to 5 (good).

- The Netflix is one of the largest benchmarks for collaborative filtering. It contains over 100 million ratings for 17,770 movies by 480,189 users. All the ratings are also ranged from 1 to 5.

We split the two datasets into training and testing sets as follows: for MovieLens, we randomly sample 80% ratings as the training set and the rest 20% is used as the testing one. As discussed in [20], in this dataset a lot of ratings are not observed, which may lead to biased evaluation results. Therefore, as [20] did, on Netflix we construct a relatively dense dataset consisting of 5,000 items with the most ratings and 10,000 users with at least 100 ratings. Then, we also sample 80% ratings and the rest as the training and testing sets respectively. For both datasets, items with ratings equal to 5 are regarded as the groundtruth items recommended to users.

In order to comprehensively evaluate the item recommendation performance using binary codes, besides MAP and PH2 we further employ Normalized Discounted Cumulative Gain (NDCG) as prior recommendation research [20] does. NDCG is widely adopted to evaluate the ranking quality in practice, mainly focusing on whether the obtained binary codes can accurately preserve the original relevance orders of the items for different users. We use NDCG calculated at different cutting points (NDCG@5 and @10) in the ranking list as the evaluation metric, and each item rating labeled by users serves as the true relevance value.

3.2.2 Results and Discussions

The proposed collaborative hashing can capture the item preferences of different users in recommendation systems, and therefore can accurately predict ratings for specific users. Traditional collaborative filtering based on matrix factorization represents users and items in a latent, low-dimensional Euclidean space, and thereby converts the item recommendation to the nearest neighbor search in such space. Instead, collaborative hashing discovers a Hamming space where entities can be encoded using compact binary codes, which largely improves the efficiency of both recommendation and storage.

In the scenario of recommendation across views (*i.e.*, users and items), we employ LCH and BCCF designed for recommendation as the baselines. Both methods outperform the standard collaborative filtering using binarized low-rank matrix factorization. Moreover, we also evaluate the traditional hashing methods including LSH and ITQ, which independently hash the entities in each view.

Table 2 shows the NDCG results on the two datasets, comparing BCCF, LCH, and naive hashing solutions using LSH and ITQ. As we can see, the NDCG calculated at different cutting points (5 and 10) increases when using more hash bits, and CH consistently achieves the best performance with up to 45.87% and 15.09% performance gains respectively on the two datasets, indicating that CH can preserve the order of user preferences very well. LSH and ITQ, only relying on the inside similarities of each view, give the worst performances in all cases. These observations demonstrate that exploring the duality between views is quite beneficial to the high-quality recommendation. Furthermore, compared to LCH and BCCF, our CH faithfully leverages the preference preservation between two views based on the neighbor patterns in each view, and thus performs best on both datasets.

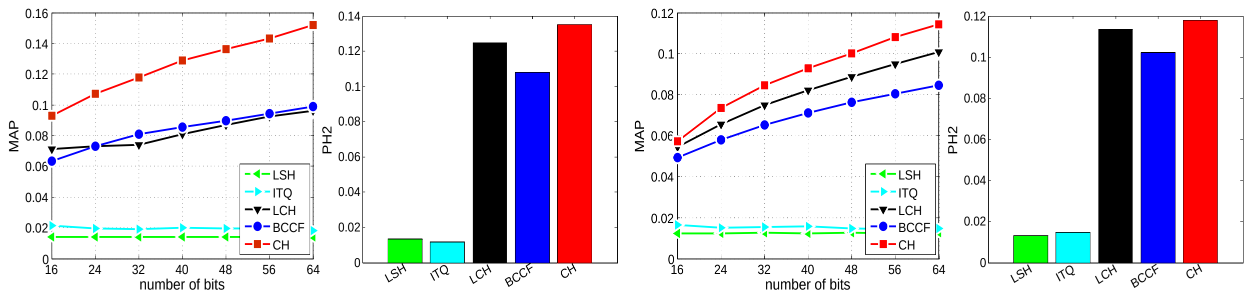
We also depict the MAP of Hamming ranking with respect to 16 - 64 hash bits and PH2 of hash lookup using 16 bits in Figure 4. Again, we can observe that the proposed hashing method ranks first with a large margin compared to the best competitors LCH and BCCF in terms of both MAP and PH2. This fact leads to the conclusion that our collaborative hashing can largely improve performances by elegantly taking advantages of the neighbor structure inside the same view and the correlations across different ones.

4. Conclusion

In this paper, we proposed a collaborative hashing scheme for data in matrix form that can learn hash codes for both types of entity in the matrix, making the proposed

Table 2. NDCG (%) of different methods using 16 - 64 bits on MovieLens and Netflix.

DATASETS	HASH	16 BITS		32 BITS		48 BITS		64 BITS	
		NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
MOVIELENS	LSH	2.99±0.22	3.00±0.23	2.89±0.13	2.88±0.09	2.90±0.14	2.91±0.13	2.93±0.22	2.90±0.20
	ITQ	4.35±0.72	4.38±0.74	3.97±0.59	3.94±0.62	4.24±0.86	4.10±0.70	4.00±0.52	3.92±0.46
	LCH	22.72±1.43	20.77±1.26	23.52±0.73	21.77±0.69	27.64±0.95	25.43±0.76	30.89±0.69	28.24±0.57
	BCCF	20.87±0.42	19.02±0.33	26.11±0.42	23.24±0.30	28.43±0.29	25.12±0.16	30.47±0.27	26.71±0.18
	CH	**28.26**±0.70	**26.39**±0.55	**36.15**±0.40	**33.09**±0.42	**41.07**±0.49	**37.28**±0.37	**45.06**±0.37	**40.64**±0.28
NETFLIX	LSH	2.57±0.16	2.57±0.16	2.69±0.21	2.65±0.18	2.67±0.20	2.65±0.19	2.62±0.12	2.63±0.10
	ITQ	3.75±0.82	3.64±0.63	3.38±0.71	3.49±0.68	3.07±0.60	3.10±0.59	3.17±0.51	3.25±0.48
	LCH	20.92±0.70	19.50±0.65	28.04±0.38	25.93±0.34	33.00±0.45	30.21±0.26	36.97±0.24	33.50±0.23
	BCCF	20.12±0.66	18.79±0.56	28.04±0.45	25.70±0.36	31.90±0.55	29.27±0.34	34.59±0.60	31.62±0.46
	CH	**23.75**±0.55	**22.25**±0.43	**33.48**±0.38	**31.02**±0.34	**38.41**±0.43	**35.45**±0.32	**42.55**±0.51	**38.95**±0.39



(a) MAP on MovieLens (b) PH2 @16 bits on MovieLens (c) MAP on Netflix (d) PH2 @16 bits on Netflix

Figure 4. Performance comparison of different methods on MovieLens and Netflix.

method conceptually unique compared with the existing methods. The key idea of the proposed method is that the learned binary codes of each type of entities should attain their neighbor structure, and meanwhile accurately preserve the correlations between different types of entities. We adopted a loss function in our optimization formulation, consisting of the binary quantization loss for each view and the deviation of predictions based on the binary codes. An efficient alternating optimization method has been proposed to learn both the hash codes and functions well preserving data distributions and the correlations. Comprehensive results over two classic applications are considerably encouraging: the proposed collaborative hashing significantly outperforms the state-of-the-art hashing methods.

Acknowledgement

This work is supported in part by NSFC 61370125 and 61101250, NCET-12-0917, SKLSDE-2013ZX-05 and SKLSDE-2014ZX-07.

References

[1] R. Arandjelović and A. Zisserman. All about VLAD. In *IEEE CVPR*, 2013. 6

[2] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280, 2007. 1, 2

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, 2004. 1, 5

[4] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011. 1, 2, 3, 4, 5, 6

[5] J. He, J. Feng, X. Liu, T. Cheng, T.-H. Lin, H. Chung, and S.-F. Chang. Mobile product search with bag of hash bits and boundary reranking. In *CVPR*, pages 3005–3012, 2012. 1

[6] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *IEEE CVPR*, pages 2938–2945, 2013. 1, 3

[7] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998. 1

[8] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, pages 304–317, 2008. 6

[9] A. Joly and O. Buisson. Random Maximum Margin Hashing. In *CVPR*, 2011. 5

[10] A. Karatzoglou, A. Smola, and M. Weimer. Collaborative filtering on a budget. In *AISTATS*, volume 9, pages 389–396, 2010. 1, 2, 5

[11] C. W. Leong and R. Mihalcea. Measuring the semantic relatedness between words and images. In *IWCS*, pages 185–194, 2011. 5

[12] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, 2012. 1, 3, 6

[13] X. Liu, J. He, B. Lang, and S.-F. Chang. Hash bit selection: A unified solution for selection problems in hashing. In *IEEE CVPR*, pages 1570–1577, 2013. 1

[14] Y. Mu, J. Shen, and S. Yan. Weakly-supervised hashing in kernel space. In *CVPR*, 2010. 1

[15] M. Norouzi and D. J. Fleet. Cartesian k-means. In *IEEE CVPR*, pages 3017–3024, 2013. 1, 3

[16] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 1, 2

[17] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *IEEE CVPR*, pages 3360–3367, 2010. 1, 2

[18] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008. 1, 4, 5

[19] D. Zhang, J. Wang, D. Cai, and J. Lu. Laplacian co-hashing of terms and documents. In *ECIR*, pages 577–580, 2010. 1, 2, 5

[20] K. Zhou and H. Zha. Learning binary codes for collaborative filtering. In *ACM SIGKDD*, pages 498–506, 2012. 1, 2, 3, 4, 5, 7