Collective Matrix Factorization Hashing for Multimodal Data

Guiguang Ding Yuchen Guo Jile Zhou
School of Software,Tsinghua University, Beijing, P.R.China
dinggg@tsinghua.edu.cn yuchen.w.guo@gmail.com jile.zip@gmail.com

Abstract

Nearest neighbor search methods based on hashing have attracted considerable attention for effective and efficient large-scale similarity search in computer vision and information retrieval community. In this paper, we study the problems of learning hash functions in the context of multimodal data for cross-view similarity search. We put forward a novel hashing method, which is referred to Collective Matrix Factorization Hashing (CMFH). CMFH learns unified hash codes by collective matrix factorization with latent factor model from different modalities of one instance, which can not only supports cross-view search but also increases the search accuracy by merging multiple view information sources. We also prove that CMFH, a similarity-preserving hashing learning method, has upper and lower boundaries. Extensive experiments verify that CMFH significantly outperforms several state-of-the-art methods on three different datasets.

1. Introduction

Nearest neighbor search plays a fundamental role in many important applications, such as information retrieval, data mining, and computer vision. Hashing-based nearest neighbor search, the most well-known method embedding high-dimensional data into compact binary codewords, has garnered considerable interest in recent years for their great efficiency gains in massive data [8]. One of the most famous hashing-based models is locality sensitive hashing (LSH) [6], whose basic idea is to map the original data into Hamming space while preserving their similarity with high probability. LSH can deal with similarity search quite efficiently because bit XOR operations are applied when calculating Hamming distance between binary codes [24]. As extensions of standard LSH, some machine learning methods are employed to design effective compact hashing, such as Manifold Learning, Supervised Learning, Kernel Learning, Deep Learning, Quantization Learning, Linear Discriminant Analysis (LDA), K-means and PCA, which respec- vised Hashing [13], Kernelized Hashing [11], Semantic Hashing [17], Iterative Quantization Hashing [7], LDA Hashing [20], K-means Hashing [9] and PCA Hashing [21].

With the spread of similarity search across different views, the single-view methods aforementioned are extended to multi-view scenarios. The core problem of hash function learning for cross-view is how to deal with multimodal data sampled from different probability distributions. Recently, a few cross-view hashing methods have been developed. Generally, cross-view hashing methods can be divided into two categories: view-specific hashing methods and integrated ones. View-specific hashing methods learn independent hash codes for each view of instance, and then concatenate multiple view-specific binary codes to obtain integrated hash codes. In [3], cross-modality similarity search hashing (CMSSH) is solved by embedding incommensurable data into a common metric space, and the hash functions are learned by using eigendecomposition and standard AdaBoost. In [12], Kumar et al. extended spectral hashing to the multi-view filed and proposed a Cross-View Hashing model (CVH), which minimizes the weighted average multi-view $\ell 2$-norm distance of object pairs by solving a generalized eigenvalue problem. Co-Regularized Hashing (CRH) [26], whose objective function intends to project data far from 0 for good generalization, and at the same time, preserve the inter-modality similarity effectively. Inter-media Hashing (IMH) [19] introduces inter-media consistency and intra-media consistency to discover a common hamming space, and uses linear regression with regularization model to learn view-specific hash functions. The aforementioned hashing methods are mainly employed in similarity search across different views. For instance, given an image as query, search engine can return some documents to accurately describe the details. To implement cross-view search, each view needs to store independent hash codes which increases the cost of storage and search.

Integrated hashing methods learn unified hash codes for each instance. Composite Hashing with Multiple Information Sources (CHMIS) [23] combines information from different sources into final integrated hash codes by optimizing the relaxed hash codes and combinatio

ternatively. Multi-View Spectral Hashing (MVSH) [10] integrates multi-view information into binary codes, and uses product of codewords to avoid undesirable embedding. This type of hashing methods is generally utilized to improve the search accuracy of hash codes by combining multiple information sources of one instance, and is not implemented for cross-view similarly search. They work well only when all the information sources are available, which is too demanding in real-world.

In this paper, we put forward a novel hashing method, which is referred to Collective Matrix Factorization Hashing (CMFH). CMFH assumes that each view of one instance generates identical hash codes, which are not the combination or concatenation of some hashing codes from different views. Figure 1 illustrates the difference among the two categories of methods aforementioned and CMFH. For each instance, we learn unified codes by collective matrix factorization with latent factor model from different view information sources. To ensure that the learn hash codes can be searched for different views, we also learn linear hash function for each view to determine binary codes of unseen instances. Our paper has the following contributions:

1. We propose a unified hashing method in cross-view scenario, which can not only support cross-view search but also increase the search accuracy by merging multiple view information.

2. Our work is the first attempt to employ the collective matrix factorization (CMF) technology to learn cross-view hash functions. Our experiments demonstrate CMF is an effective hashing method when multiple view information sources are available.

3. We show that the proposed CMFH is a similarity-preserving hashing method with approximate bi-Lipschitz continuity as illustrated in 3.6.

Our extensive experimental study on three different datasets highlights the advantage of our method under cross-view scenarios and verifies that CMFH significantly outperform several state-of-the-art methods.

The remainder of this paper is organized as follows. We briefly introduce the related work on collective matrix factorization and representative cross-view hashing methods in Section 2. Section 3 presents our proposed approach. Section 4.1 provides extensive experimental validation on three datasets. The conclusions are given in Section 5.

2. Related Work

2.1. Collective Matrix Factorization

For relational learning, [18] proposed Collective matrix factorization to predict unknown values of relation given an entity dataset and observed multiple relations among

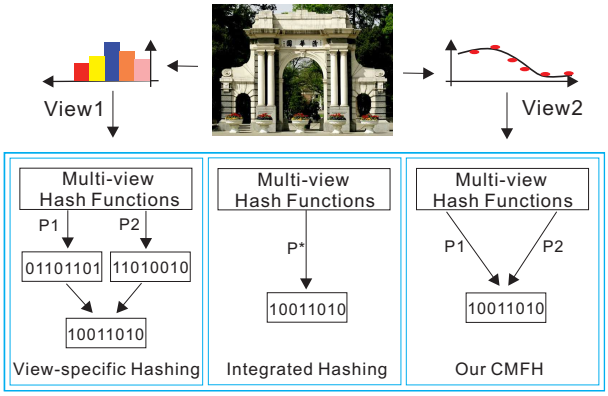

Figure 1. The difference among the three types of methods.

entities. CMF jointly factorizes multiple relation matrices which may have different value types, and the factors share parameters when entities appear in multiple relations. Take movie rating prediction as an example in [18]. Let $X \in \mathbb{R}^{m\times n}$ be an integer matrix representing user's rating, and the element X_{ij} denotes user i's rating for movie j. Let $Y \in \mathbb{R}^{r\times n}$ be a binary matrix denoting the genres each movie belonging to, and Y_{ij} indicate whether movie j belongs to genre i or not. The factors are $U \in \mathbb{R}^{m\times k}, V \in \mathbb{R}^{n\times k}$ and $Z \in \mathbb{R}^{r\times k}$, and V is the shared factor in both reconstructions: $X \approx f_1(UV^T)$ and $Y \approx f_2(ZV^T)$, where f_i is a possibly-nonlinear link function, and $k > 0$ is the rank. The average decomposable losses are:

$$L(U, V, Z|X, Y) = \alpha_1 L_1(U, V|X) + \alpha_2 L_2(Z, V|Y)$$

where L_1, L_2 is decomposable loss function for $X \approx f_1(UV^T)$ and $Y \approx f_2(ZV^T)$ respectively, and $\sum_t \alpha_t = 1$.

Not limited to movie rating prediction, CMF is a simple yet powerful approach to deal with many applications, where multiple interlinked sources of data are available and they cannot be represented by a single adjacency matrix [2]. In multimedia domain, objects are often presented in several different views, *e.g.* Wikipedia's pages of the same topic may emerge in the forms of image, text or both. To our best knowledge, we are the first to apply CMF to learn hashing functions for similarity search on multimodal data.

2.2. View-specific Hashing Methods

View-specific hashing methods learn independent hash codes for each view of instance, and then concatenate view-specific binary codes for obtaining integrated hash codes.

CVH [12] designs a set of binary codes $\{\mathbf{y}_i^{(t)}\}$ for view t of object $o_i, \forall t, i$, and minimises the weighted cumulative Hamming distance:

$$\begin{aligned} &\text{minimize} \quad d = \sum_{ij} \mathbf{W}_{ij} d_{ij} \\ &\text{s.t.} \quad \mathbf{y}_i^{(t)} \in \{-1, 1\}, \sum_i \mathbf{y}_i^{(t)} = 0, \frac{1}{n} \sum_i \mathbf{y}_i^{(t)} \mathbf{y}_i^{(t)^T} = I, \forall t \end{aligned} \tag{1}$$

Where $d_{ij} = \sum_t \sum_{t' \geq t} ||\mathbf{y}_i^{(t)} - \mathbf{y}_j^{(t')}||^2$ denotes the cumulative Hamming distance between o_i and o_j, and $\mathbf{W}_{ij}$ be the similarity between o_i and o_j. At last, a low-dimensional linear embedding is assumed: $\mathbf{y}_i^{(t)} = {A^{(t)}}^T \mathbf{x}_i^{(t)}$, which transforms Equation (1) to a generalized eigenvalue problem.

CMSSH [3] embeds the input data from two arbitrary spaces into the Hamming space in a supervised way. Given pair $(\mathbf{x}_k, \mathbf{y}_k)$ and similarity label $s_k \in \{+1, -1\}$, here $\mathbf{x}_k \in \mathbb{R}^m, \mathbf{y}_k \in \mathbb{R}^n$ are sampled from different spaces. CMSSH defines the affine projections of form $f_i(\mathbf{x}) = \mathbf{p}_i^T \mathbf{x} + a_i$ and $g_i(\mathbf{y}) = \mathbf{q}_i^T \mathbf{y} + b_i$, then the i-th bit is generated by maximizing:

$$r_i = \sum_k w_i(k) s_k \text{sign}(\mathbf{p}_i^T \mathbf{x}_k + a_i) \text{sign}(\mathbf{q}_i^T \mathbf{y}_k + b_i) \quad (2)$$

where $w_i(k)$ is weighting coefficients in Adaboost [1], and $w_i(k)$ is increased for (x_k, y_k) that is misclassified and decreased otherwise. Discarding the sign function, Equation (2) is closely related to a simpler function:

$$\widehat{r}_i = \sum_k v_k (\mathbf{p}_i^T \overline{\mathbf{x}}_k)(\mathbf{q}_i^T \overline{\mathbf{y}}_k) = \mathbf{p}_i^T (\sum_k v_k \overline{\mathbf{x}}_k \overline{\mathbf{y}}_k^T) \mathbf{q}_i \quad (3)$$

where $\overline{\mathbf{x}}_k$ and $\overline{\mathbf{y}}_k^T$ are x_k and y_k centered by their weighted means, and $v_k = w_i(k) s_k$. Equation (3) can be solved by singular value decomposition, and multiple hash bits can be generated by Adaboost framework.

2.3. Integrated Hashing methods

Integrated hashing methods learn only a set of unified hash codes for each instance. CHMIS [23] uses a codeword $\mathbf{y}_i^*$ to represent an object o_i, and measures the Hamming distance on each individual source and sum them up: $\sum_t \sum_{ij} \mathbf{W}_{ij}^{(t)} \|\mathbf{y}_i^* - \mathbf{y}_j^*\|^2$, where $\mathbf{W}^{(t)}$ is the affinity matrix for t-th source. In order to extend the solution to out-of-sample datapoints easily, CHMIS assumes that $\mathbf{y}_i^* = \text{sign}(\sum_t \alpha_t (\mathbf{M}^{(t)}) \mathbf{x}_i^{(t)})$, where $\mathbf{M}^{(t)}$ is the weight matrix for the t-th source, and $\alpha = \{\alpha_t\}$ is a non-negative convex combination with $\sum_t \alpha_t = 1$. CHMIS minimises:

$$C_1 \sum_t \sum_{ij} \mathbf{W}_{ij}^{(t)} \|\mathbf{y}_i^* - \mathbf{y}_j^*\|^2$$
$$+ C_2 \sum_i \|\mathbf{y}_i^* - \sum_t \alpha_t (\mathbf{M}^{(t)}) \mathbf{x}_i^{(t)}\|^2 + \lambda \sum_t \|\mathbf{M}^{(t)}\|^2$$

MVSH [10] constructs an average similarity matrix $\mathbf{W}$, and integrates multiple view information to unified binary codes as $\mathbf{y}_i^* = \text{sign}(\sum_t \mathbf{M}^{(t)} \mathbf{x}_i^{(t)})$, where $\mathbf{M}^{(t)}$ is the projection matrix. In order to avoid undesirable embedding, MVSH defines $d_{ij} = (\mathbf{y}_i^*)^T \mathbf{y}_j^*$ to measure Hamming distance between o_i and o_j. Hence MVSH minimises:

$$\sum_{ij} \mathbf{W}_{ij} (\sum_t \mathbf{M}^{(t)} \mathbf{x}_i^{(t)})^T (\sum_t \mathbf{M}^{(t)} \mathbf{x}_i^{(t)})$$

3. Collective Matrix Factorization Hashing

In this section, we present our hashing method for multiple modalities data, *i.e*. Collective Matrix Factorization Hashing(CMFH). Without loss of generality, we introduce CMFH firstly in bimodal case because it is simple and easy to understand.

3.1. Problem Formulation

Suppose that $\mathcal{O} = \{o_i\}_{i=1}^n$ is a set of multi-view objects and $\mathbf{X}^{(1)} = [\mathbf{x}_1^{(1)}, ..., \mathbf{x}_n^{(1)}]$, $\mathbf{X}^{(2)} = [\mathbf{x}_1^{(2)}, ..., \mathbf{x}_n^{(2)}]$ are two different view matrices of $\mathcal{O}$, where $\mathbf{x}_i^{(1)} \in \mathbb{R}^{d_1}$, $\mathbf{x}_i^{(2)} \in \mathbb{R}^{d_2}$ (usually, $d_1 \neq d_2$). Given the codewords length k, the purpose of CMFH is to learn unified hash codes $\mathbf{y}_i \in \{-1, 1\}^k$ for $o_i, i = 1, 2, ..., n$, such that $\mathbf{y}_i, \mathbf{y}_j$ preserve the similarity between o_i and o_j with high probability.

3.2. Framework Overview

As illustrated in Figure 2, the proposed CMFH consists of two phases. One is offline hash functions learning and datebases generating, the other is online coding and searching. In offline phase, CMFH learns unified hash codes $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_n]$. For out-of-sample instance, CMFH learns view-specific hash functions f_t for t-th view. Similar to previous work [21, 12, 3], we only consider the affine projections of the form

$$f_t(\mathbf{x}^{(t)}) = \mathbf{P}_t \mathbf{x}^{(t)} + \mathbf{a}_t, \forall t \quad (4)$$

where $\mathbf{P}_t \in \mathbb{R}^{k \times d_t}$ is the projection matrix, and $\mathbf{a}_t \in \mathbb{R}^k$ is the offset unit vector. In the online phase, queries of any type would be mapped to compact codes according to related learned hashing functions, *i.e*. given the query $\mathbf{x}^{(t_0)}$, generating unified hashwords by $\mathbf{y} = \text{sign}(f_{t_0}(\mathbf{x}^{(t_0)}))$. Then CMFH returns similar results of all views for the given mapped query. CMFH is quite efficient for online similarity search task, since bit XOR operations are applied when calculating Hamming distance between binary codes.

3.3. Collective Matrix Factorization Hashing

We can learn latent semantic feature from source datasets by matrix factorization [5]:

$$\mathbf{X}^{(t)} = \mathbf{U}_t \mathbf{V}_t, \forall t \quad (5)$$

where $\mathbf{U}_t \in \mathbb{R}^{d_t \times k}, \mathbf{V}_t \in \mathbb{R}^{k \times n}$, and k is the number of latent factors. Each column vector $\mathbf{v}_t$ is a latent semantic representation of the t-th view data $\mathbf{x}^{(t)}$. It is assumed that:

1. The interlinked data should have the same latent semantic representation;

2. The hash codes can be learned from latent semantic representation, *i.e*. $\mathbf{y} = \text{sign}(\mathbf{v})$.

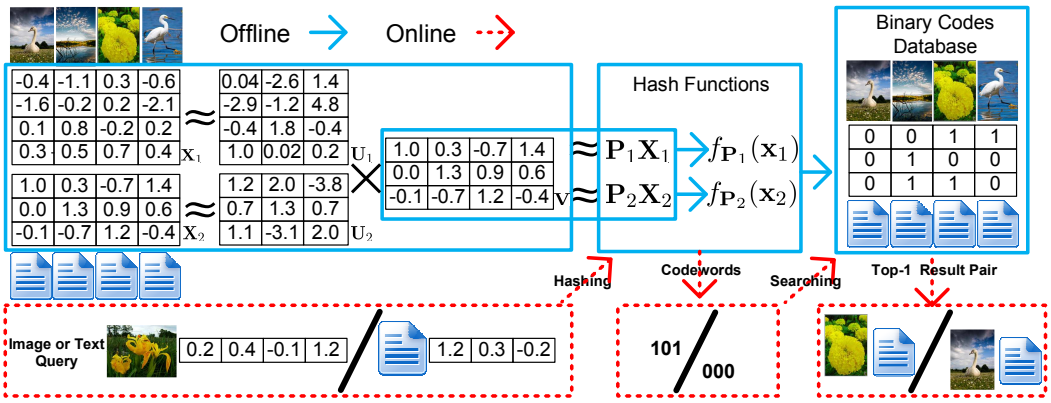

Figure 2. Framework of CMFH, illustrated with toy data.

Based on assumption 1, we decompose $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ jointly with the constraint $\mathbf{V}_1 = \mathbf{V}_2 = \mathbf{V}$:

$$\lambda||\mathbf{X}^{(1)} - \mathbf{U}_1\mathbf{V}||_F^2 + (1-\lambda)||\mathbf{X}^{(2)} - \mathbf{U}_2\mathbf{V}||_F^2 \quad (6)$$

Here λ is the balance parameter. The $\mathbf{Y}$ for database can be obtained directly based on assumption 2, but it cannot be generalized to query straightly. For out-of-sample instance, CMFH learns view-specific hash functions f_t for t-th view of the form in Equation (4). A balanced hash function, which meets $\sum_i \text{sign}(f_t(\mathbf{x}_i^{(t)})) = 0$, would maximum information on $\mathbf{X}^{(t)}$ [21]. Dropping sign function, the balanced constraint would lead to $\mathbf{a}_t = -\sum_i \mathbf{P}_t\mathbf{x}_i^{(t)}/n$, then we can rewrite Equation (4) as $f_t(\mathbf{x}^{(t)}) = \mathbf{P}_t(\mathbf{x}^{(t)} - \sum_i \mathbf{x}_i^{(t)}/n) = \mathbf{P}_t\overline{\mathbf{x}}^{(t)}$. However, we still use $\mathbf{x}^{(t)}$ to denote the centred data $\overline{\mathbf{x}}^{(t)}$ for convenience.

The overall objective function combines the collective matrix factorization part given in Equation (6), the linear embedding part in Equation (4) and regularization term:

$$\underset{\mathbf{U}_1,\mathbf{U}_2,\mathbf{P}_1,\mathbf{P}_2,\mathbf{V}}{\text{minimise}} \; G(\mathbf{U}_1,\mathbf{U}_2,\mathbf{P}_1,\mathbf{P}_2,\mathbf{V}) \quad (7)$$

where

$$\begin{aligned} G =& \lambda\|\mathbf{X}^{(1)} - \mathbf{U}_1\mathbf{V}\|_F^2 + (1-\lambda)\|\mathbf{X}^{(2)} - \mathbf{U}_2\mathbf{V}\|_F^2 \\ &+\mu(\|\mathbf{V} - \mathbf{P}_1\mathbf{X}^{(1)}\|_F^2 + \|\mathbf{V} - \mathbf{P}_2\mathbf{X}^{(2)}\|_F^2) \\ &+\gamma R(\mathbf{V},\mathbf{P}_1,\mathbf{P}_2,\mathbf{U}_1,\mathbf{U}_2) \end{aligned} \quad (8)$$

where μ and γ are tradeoff parameters, and regularization term is defined as $R(\cdot) = ||\cdot||_F^2$ to avoid overfitting.

3.4. Learning Hash Function

The optimization problem (7) is non-convex with five matrix variables $\mathbf{U}_1, \mathbf{U}_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{V}$. Fortunately, it is convex with respect to any one of the five matrix variables while fixing the other four. Therefore, the optimization problem can be solved by following the listed three steps iteratively until convergency:

1. Fix $\mathbf{P}_t, \mathbf{V}$, let $\frac{\partial G}{\partial \mathbf{U}_t} = 0, t = 1, 2$, then obtain:

$$\mathbf{U}_t = \mathbf{X}^{(t)}\mathbf{V}^T(\mathbf{V}\mathbf{V}^T + \frac{\gamma}{\lambda_t}\mathbf{I})^{-1} \quad (9)$$

where $\lambda_1 = \lambda, \lambda_2 = 1-\lambda$, and $\mathbf{I}$ is the identity matrix.

2. Fix $\mathbf{U}_t, \mathbf{V}$, let $\frac{\partial G}{\partial \mathbf{P}_t} = 0, t = 1, 2$, then obtain:

$$\mathbf{P}_t = \mathbf{V}_t{\mathbf{X}^{(t)}}^T(\mathbf{X}^{(t)}{\mathbf{X}^{(t)}}^T + \frac{\gamma}{\mu}\mathbf{I})^{-1} \quad (10)$$

3. Fix $\mathbf{U}_t, \mathbf{P}_t$, let $\frac{\partial G}{\partial \mathbf{V}} = 0, t = 1, 2$, then obtain:

$$\mathbf{V} = (\sum_{t=1}^{2}\lambda_t\mathbf{U}_t^T\mathbf{U}_t + (2\mu+\gamma)\mathbf{I})^{-1}(\sum_{t=1}^{2}(\lambda_t\mathbf{U}_t^T + \mu\mathbf{P}_t)\mathbf{X}^{(t)}) \quad (11)$$

The algorithm is summarized in Algorithm 1.

Algorithm 1 Collective Matrix Factorization Hashing

Input:

Data matrix $\mathbf{X}^{(t)}, t = 1, 2$, parameters λ, μ, γ, k

Output:

Integrated hash codes $\mathbf{Y}$, projection matrix $\mathbf{P}_t, t = 1, 2$.

Initialize $\mathbf{U}_t, \mathbf{P}_t$ by random matrices, and centering $\mathbf{X}^{(t)}$ by means, $t = 1, 2$.
repeat
  Fix $\mathbf{U}_t, \mathbf{P}_t$, update $\mathbf{V}$ by Equation (11),$t = 1, 2$;
  Fix $\mathbf{U}_t, \mathbf{V}$, update $\mathbf{P}_t$ by Equation (10),$t = 1, 2$;
  Fix $\mathbf{P}_t, \mathbf{V}$, update $\mathbf{U}_t$ by Equation (9),$t = 1, 2$;
until convergency.
$\mathbf{Y} = \text{sign}(\mathbf{V})$.

3.5. Learning Hash Codes

CMFH first utilizes Equation (11) to compute $\mathbf{V}$, then generates hashcodes for the whole database by a simple thresholding strategy, *i.e.* sign($\mathbf{V}$). When a new query $\widetilde{\mathbf{x}}^{(t)}$ comes, CMFH generates the hashcodes according to Equation (4): $\widetilde{\mathbf{y}}^{(t)} = \text{sign}(\mathbf{P}_t(\widetilde{\mathbf{x}}^{(t)} - \sum_i \mathbf{x}_i^{(t)}/n))$.

3.6. Theoretical Analysis

CMFH is a similar-preserving hashing method. According to formula (7), for every view t and every instance i:

$$\begin{aligned} \mathbf{x}_i^{(t)} &= \mathbf{U}_t \mathbf{v}_i + \mathbf{e}_i^{(t)} \\ \mathbf{v}_i &= \mathbf{P}_t \mathbf{x}_i^{(t)} + \mathbf{e'}_i^{(t)} \end{aligned} \tag{12}$$

where $\mathbf{e}_i^{(t)}, \mathbf{e'}_i^{(t)}$ are the reconstruction errors. Based on Equation (12), the norm of instance i minus instance j is:

$$\begin{aligned} \|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)} - (\mathbf{e}_i^{(t)} - \mathbf{e}_j^{(t)})\| &= \|\mathbf{U}_t(\mathbf{v}_i - \mathbf{v}_j)\| \\ \|\mathbf{v}_i - \mathbf{v}_j\| &= \|\mathbf{P}_t(\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}) + (\mathbf{e'}_i^{(t)} - \mathbf{e'}_j^{(t)})\| \end{aligned} \tag{13}$$

Using the norm properties $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$ and $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$ for any matrix $\mathbf{A}, \mathbf{B}$, then obtaining:

$$\begin{aligned} \|\mathbf{v}_i - \mathbf{v}_j\| &\geq C_1 \|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\| - \epsilon_1^{(t)}(i,j) \\ \|\mathbf{v}_i - \mathbf{v}_j\| &\leq C_2 \|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\| + \epsilon_2^{(t)}(i,j) \end{aligned} \tag{14}$$

where $C_1 = \max_t\{1/\|\mathbf{U}_t\|\}, C_2 = \min_t\{\|\mathbf{P}_t\|\}$, and the bound errors $\epsilon_1^{(t)}(i,j) = \|\mathbf{e}_i^{(t)} - \mathbf{e}_j^{(t)}\|/\|\mathbf{U}_t\|, \epsilon_2^{(t)}(i,j) = \|\mathbf{e'}_i^{(t)} - \mathbf{e'}_j^{(t)}\|$. We denote $\epsilon_1 = \{\epsilon_1^{(t)}(i,j); \forall t, i, j\}$ and $\epsilon_2 = \{\epsilon_2^{(t)}(i,j); \forall t, i, j\}$ as bound error set. The inequalities in (14) explain the bounds of $\|\mathbf{v}_i - \mathbf{v}_j\|$, which means that the hash functions of CMFH are approximate bi-Lipschitz continuity [1]. However, the value of ϵ_t affects the local sensitivity of CMFH significantly . If $\mathbf{X}^{(t)}$ and $\mathbf{V}$ are decomposed perfectly, then the error term ϵ_t equates to 0, and the bounds in (14) guarantee $\mathbf{v}_i$ tends to $\mathbf{v}_j$ when $\mathbf{x}_i^{(t)}$ tends to $\mathbf{x}_j^{(t)}$. In fact, minimising (7) would reduce the value of ϵ_t (usually, not equates to 0).

We investigate the distribution of ϵ_t among one large dataset. We train 64-bits CMFH firstly, then 5000 instances are sampled to compute reconstruction error by Equation (12). In order to eliminate the influence of data dimension, $\mathbf{e}_i^{(t)}/\|\mathbf{v}_i\|$ and $\mathbf{e'}_i^{(t)}/\|\mathbf{x}_i^{(t)}\|$ are used as normalized reconstruction error. Then bound errors are computed based on them. Therefore, we draw the distribution histogram among total 25M data pairs in Figure 3. More than 90% data pairs of ϵ_1 and ϵ_2 fall into $[0, 0.1]$, which means that the bounds of CMFH in (14) is tight in practice.

The time complexity for training CMFH is $O((d^3 + k^3 + nkd + k^2n + dk^2 + n^2d + n^2k)T)$, where T is the number of iterations, and $d = \max_t\{d_t\}$. Because $k, d \ll n$, the overall complexity is $O(n^2(d+k)T)$. In fact, training CMFH is faster than most existing multi-view hashing methods. For each query, the hashing time is $O(dk)$, which is identical to that of matrix multiplication.

[1]The inequalities (14) is similar with bi-Lipschitz continuity in *mathematical analysis* if dropping bound error terms, so we call (14) as approximate bi-Lipschitz continuity.

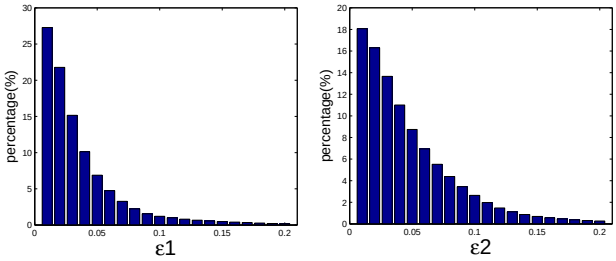

Figure 3. The histogram distribution of bound errors.

3.7. Extension

The extension for CMFH in formula (7) from bimodal to multiple modalities is quite simple and direct:

$$\begin{aligned} \underset{\mathbf{U}_t, \mathbf{P}_t, \mathbf{V}}{\text{minimise}} & \sum_t \lambda_t ||\mathbf{X}^{(t)} - \mathbf{U}_t \mathbf{V}||_F^2 + \mu \sum_t ||\mathbf{V} - \mathbf{P}_t \mathbf{X}^{(t)}||_F^2 \\ & + \gamma(\sum_t R(\mathbf{P}_t, \mathbf{U}_t) + R(\mathbf{V})) \end{aligned}$$

where $\sum_t \lambda_t = 1$. It is straightforward to adapt Algorithm 1 presented above to solve the new problems.

4. Experiments

We carried out our experiments on three different datasets : Wiki, NUS-WIDE and MIRFLICKR-25000. We compared the proposed CMFH to several state-of-the-art methods and the experiment results show that CMFH can significantly outperform the baseline methods.

4.1. Experiment Settings

4.1.1 Datasets

Wiki[2]. It was collected from Wikipedia with 2,866 image-text pairs. Each image is represented by 128-dimension SIFT [15] histograms and each text is represented by 10-dimension topics vector. It contains 10 semantic classes and each pair is labeled with one of them. We use 75% of the pairs as the training set, the remaining 25% as the query set.

NUS-WIDE[3]. It is a real-world web image database containing 81 concepts and 269,648 images with tags. We select ten largest concepts and the corresponding 186,577 images. Images are represented by 500-dimension SIFT histograms, and texts are represented by index vectors of the most frequent 1000 tags. Each pair is annotated by at least one of 10 concepts. Pairs are considered to be similar if they share at least one concepts. We use 99% of the data as the training set and the rest 1% as the query set.

MIRFLICKR-25000[4]. It consists of 25,000 images, and each image is annotated by some labels in 38 unique

[2]http://www.svcl.ucsd.edu/projects/crossmodal/
[3]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm
[4]http://press.liacs.nl/mirflickr/

Table 1. mAP Comparison, View1 is Image or CEDD, and View2 is Text or SIFT.

Task	Method	Wiki				NUS-WIDE				MIRFLICKR-25000			
		16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
View1 to View2	CVH	0.2041	0.1604	0.1296	0.1308	0.3722	0.3632	0.4060	0.3875	0.6070	0.5999	0.5911	0.5778
	IMH	0.2017	0.2119	0.2025	0.1926	0.4721	0.4716	0.4668	0.4594	0.6124	0.6034	0.5930	0.5795
	CMSSH	0.2026	0.2113	0.2011	0.2084	0.4997	0.5189	0.5142	0.5086	0.5417	0.5438	0.5406	0.5486
	CHMIS	0.2207	0.2189	0.2126	0.2035	0.4870	0.4896	0.4835	0.4762	0.6227	0.6202	0.6147	0.6087
	CMFH	**0.2538**	**0.2582**	**0.2619**	**0.2648**	**0.5591**	**0.5698**	**0.5780**	**0.5837**	**0.6480**	**0.6597**	**0.6693**	**0.6752**
View2 to View1	CVH	0.2962	0.1944	0.1337	0.1125	0.4042	0.3991	0.4480	0.4315	0.6095	0.6040	0.5955	0.5822
	IMH	0.4865	0.5295	0.4935	0.4628	0.4793	0.4800	0.4744	0.4643	0.6046	0.5983	0.5866	0.5771
	CMSSH	0.2928	0.2732	0.2753	0.2814	0.5015	0.5103	0.5039	0.4984	0.6158	0.6206	0.6194	0.6213
	CHMIS	0.2207	0.2189	0.2126	0.2035	0.4870	0.4896	0.4835	0.4762	**0.6227**	0.6202	0.6147	0.6087
	CMFH	**0.6116**	**0.6298**	**0.6398**	**0.6477**	**0.6614**	**0.6921**	**0.7164**	**0.7185**	0.6174	**0.6241**	**0.6311**	**0.6340**

labels. Images are described by 100-dimension SIFT [15] histograms which mainly encode surface texture, and 144-dimension CEDD features [4] which focus on color and edge directivity. Because they are similar in texture space while dissimilar in color space, we choose them to simulate a cross-view setting as in [16]. We use 75% of the pairs as the training set, and the remaining 25% as the query set.

4.1.2 Baseline Methods

CMFH is compared against five state-of-the-art hashing methods: LSH[5], CVH[5], IMH[5], CMSSH[6] and CHMIS[6]. They can be divided into three types. LSH is a single-view hashing method, while CVH, IMH and CMH are cross-view hashing methods in which view-specific hash codes are learned, and CHMIS is a cross-view method which learns integrated hash codes. We carefully tuned the their parameters and reported their best results.

4.1.3 Evaluation Metric

The performance measure is the mean average precision (mAP), which is the mean of average precision (AP), and AP of top R retrieved instances is defined as:

$$\text{AP} = \frac{1}{L}\sum_{i=1}^{R} P(i) \times \delta(i) \tag{15}$$

where L is the number of relevant instances in retrieved set, $P(i)$ denotes the precision of the top i retrieved documents, and $\delta(i)$ is an indicator function equaling 1 if the item at rank i is a relevant document, 0 otherwise.

We also report *precision-recall* curves on Wiki dataset. It can be obtained by varying the Hamming radius of the retrieved points and evaluating the precision, recall and the number of retrieved points accordingly.

4.1.4 Implementation Details

IMH and CHMIS require so much computational resource that it's quite difficult to learn hash functions from the whole set on NUS-WIDE and MIRFLICKR-25000. Thus, we select 5000 instances from database randomly as the training set to learn hash functions. Then hash functions are applied to every instance in database to obtain its hash codes.

In the coming sections, we provide empirical analysis on parameter sensitivity, which verifies that CMFH can achieve stable performance under a wide range of parameter values. When comparing with the baseline methods, we use the following parameter settings: $\lambda = 0.5$, $\mu = 100$, $\gamma = 0.01$ and $R = 50$. To remove any randomness caused by random initialization and random selection of training set, all of the results are averaged over 25 runs.

4.2. Results and Discussions

4.2.1 Results on Wiki

The mAP values for CMFH and four baseline methods are reported in Table 1. The *precision-recall* curves are plotted in Figure 4. We can observe that CMFH significantly outperforms all baseline methods on both tasks varying code length, which verifies the effectiveness of CMFH.

Furthermore, CMFH performs better with longer codes. This is reasonable because longer hash codes can encode more information and thus can improve the mAP performance. However, We can observe that the PR-curve of several methods looks strange, e.g. the PR-curve of CVH at 64 bits shows that it behave like random guess in experiments. This phenomenon has also been observed in [25] [14]. Actually, all baseline methods are solved by eigenvalue decomposition and have orthogonality constraints on each bit so that each bit shows no correlation to each other. The first few projection directions may have high variance and their corresponding hash bits can be quite discriminative, which is quite useful to similarity search. However, as the code length increases, the hash codes will be dominated by bits with very low variance. Actually, since the variance is too low, the lower bits are meaningless and ambiguous. So these indiscriminative hash bits may lead the method to make random guess in experiments.

[5]We implemented it because the code is not publicly available.
[6]The source code is kindly provided by the authors.

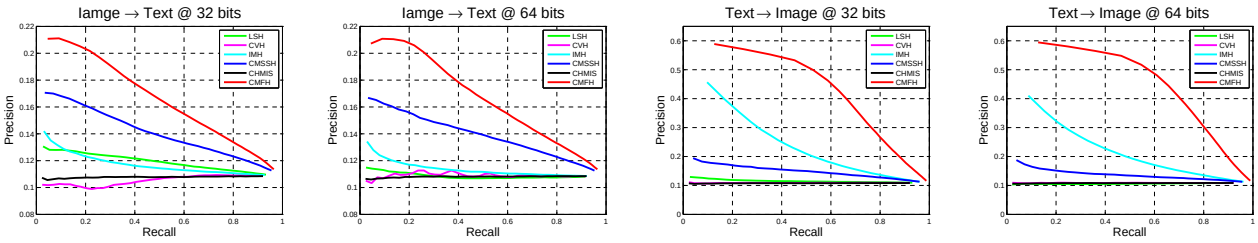

Figure 4. PR-Curves on Wiki Varying Code Length

4.2.2 Results on NUS-WIDE

The mAP values for CMFH and four baseline methods are reported in Table 1. As mentioned above, we select 5000 instances randomly as the training set to learn hash functions. And the learned hash functions are extended to the whole database. CMFH significantly outperforms all baseline methods on both tasks varying code length.

In addition, we compare the performance of different hashing methods for single-view similarity search on NUS-WIDE. We can observe that CMFH, which combines multiple information sources of one instance, can outperform other hashing methods. This is reasonable because more information of one instance can be encoded into hash codes when combining multiple views. Furthermore, this result also validates the ability of CMFH to increase the search accuracy by merging multiple information sources.

In real-world applications, the size of database may be so large that it's impossible to learn hash functions on the whole database because of the limitation of computational resource. And new data is keeping coming into database as time goes by and hash codes for new data must be computed. One solution to these problems is to learn hash functions on a smaller training set and extend it to out-of-sample instances (other instances in database or new-coming instances). The experiment settings on NUS-WIDE is quite similar to real-world scenario. The experiment results show that CMFH can handle out-of-sample instances easily and the ability to handle large-scale database.

4.2.3 Results on MIRFLICKR-25000

The mAP values for CMFH and four baseline methods for cross-view similarity search are reported in Table 1. As the results above, CMFH can outperform all baseline methods varying code length for cross-view similarity search.

4.2.4 Parameter Sensitivity

We conduct empirical analysis on parameter sensitivity on all six tasks. μ controls the weight of linear embedding. If it's too large, Equation (7) is equivalent to maximize inter-correlation as a special case of CVH which will leads to poor performance. If it's too small, the hash functions can't preserve locality as it loose the upper bound. γ controls the complexity of the model. The model is under-fitted with too large value while over-fitted with too small value.

Table 2. mAP Comparison on NUS-WIDE for Single-view Similarity Search

Task	Method	Code Length			
		16	32	64	128
Img to Txt	LSH	0.4362	0.4490	0.4651	0.4794
	CVH	0.4662	0.4689	0.4807	0.4868
	IMH	0.4821	0.4835	0.4837	0.4868
	CMSSH	0.4819	0.4866	0.4910	0.4936
	CHMIS	0.4870	0.4896	0.4835	0.4762
	CMFH	**0.5591**	**0.5698**	**0.5780**	**0.5837**
Txt to Img	LSH	0.5004	0.5758	0.6588	0.7110
	CVH	0.4193	0.3937	0.4934	0.4729
	IMH	0.5634	0.6444	0.6942	0.7120
	CMSSH	0.6008	0.6332	0.6440	0.6549
	CHMIS	0.4870	0.4896	0.4835	0.4762
	CMFH	**0.6614**	**0.6921**	**0.7164**	**0.7185**

Table 3. Effect of Training Size on mAP

DataSet	Task	Training Size		
		5k	10k	15k
NUS-WIDE	Img to Txt	0.5651	0.5785	0.5871
	Txt to Img	0.7164	0.7306	0.7384
MIRFLICKR	CEDD to SIFT	0.6693	0.6736	0.6764
	SIFT to CEDD	0.6311	0.6348	0.6383

The code length is fixed to 64, and other parameters are set as introduced in 4.1.4. We conduct analysis on one parameter by varying its value while fixing the other. The results, plotted in Figure 5 where View1 is Image or CEDD and View2 is Text or SIFT, validate that CMFH can achieve fantastic performance under a wide range of parameter values. The dashed lines show the best baseline results. On all six tasks, CMFH outperforms best baseline results when $\mu \in [1, 1000]$ and $\gamma \in [0.001, 0.1]$.

4.2.5 Effect of Training Size

Furthermore, we study the effect of size of training set on NUS-WIDE and MIRFLICKR-25000. Table 3 shows the mAP results varying the size of training data from 5k to 15k. Clearly, it's expected that more effective hash functions can be learned given more data. However, as the size of training data increase to 5k, further increasing training data size does'nt significantly improve the results. This shows the stabilization of the hash functions learned by CMFH with reasonably small training set.

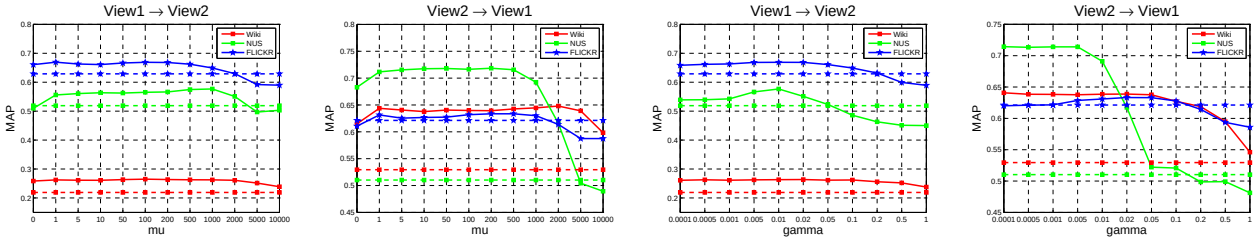

Figure 5. Parameter Sensitivity Analysis (μ and γ)

5. Conclusions

In this paper, we propose a novel hashing method, referred to Collective Matrix Factorization Hashing, for cross-view similarity search on multimodal data. CMFH learns unified hash codes by collective matrix factorization with latent factor model from different modalities of one instance, which can be searched for different views. We also show that CMFH is a similarity- preserving hashing method with approximate bi- Lipschitz continuity.

We conduct experiments to verify the effectiveness of the proposed CMFH. We show that CMFH achieves much better performance than several state-of-the-art hashing methods in all cross-view and most single-view similarity search experiments. The parameter analysis shows that CMFH is not sensitive to parameter settings, which can deliver remarkable performance under a wide range of parameter values. Furthermore, CMFH is capable of dealing with out-of-sample instances easily and can learn stable hash functions with reasonably small training set from a large-scale database, which makes it applicable to real-world scenarios.

6. Acknowledgments

This research was supported by the National Basic Research Project of China (Grant No. 2011CB70700), the National Natural Science Foundation of China (Grant No. 61271394), and the National HeGaoJi Key Project (No. 2013ZX01039-002-002). And the authorswould like to thank the reviewers for their valuable comments.

References

[1] C. M. Bishop et al. *Pattern recognition and machine learning*. Springer, New York, 2006.

[2] G. Bouchard, S. Guo, and D. Yin. Convex collective matrix factorization. In *AISTATS*, 2013.

[3] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*. IEEE, 2010.

[4] S. A. Chatzichirstofis and Y. S. Boutalis. Cedd: color and edge directivity descriptor: a compact descriptor for image indexing and retrieval. In *Computer Vision Systems*, 2008.

[5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 1990.

[6] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[7] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*. IEEE, 2011.

[8] 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*. IEEE, 2012.

[9] K. He, F. Wen, and J. Sun. K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013.

[10] S. Kim, Y. Kang, and S. Choi. Sequential spectral learning to hash with multiple representations. In *ECCV*. Springer, 2012.

[11] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*. IEEE, 2009.

[12] S. Kumar and R. Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*. AAAI Press, 2011.

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

[14] W. Liu, J. Wang, S. Kumar, and S. F. Chang. Hashing with graphs. In *ICML*, 2011.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[16] N. Quadrianto and C. H. Lampert. Learning multi-view neighborhood preserving projections. In *ICML*, 2011.

[17] R. Salakhutdinov and G. Hinton. Semantic hashing. *IJAR*, 2009.

[18] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *SIGKDD*. ACM, 2008.

[19] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *ICMD*. ACM, 2013.

[20] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *PAMI*, 2012.

[21] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*. IEEE, 2010.

[22] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. NIPS, 2008.

[23] D. Zhang, F. Wang, and L. Si. Composite hashing with multiple information sources. In *SIGIR*. ACM, 2011.

[24] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *SIGIR*. ACM, 2010.

[25] Y. Zhen and D. Yang. A probabilistic model for multimodal hash function learning. In *SIGKDD*, 2012.

[26] Y. Zhen and D.-Y. Yeung. Co-regularized hashing for multimodal data. In *NIPS*, 2012.