Coordinate Discrete Optimization for Efficient Cross-View Image Retrieval

Yadong Mu[1], **Wei Liu**[2], **Cheng Deng**[3], **Zongting Lv**[2], **Xinbo Gao**[3]
[1]Institute of Computer Science and Technology, Peking University, Beijing, 100871, China
[2]Didi Research, Beijing, 100193, China [3]Xidian University, Shannxi, 710126, China

Abstract

Learning compact hash codes has been a vibrant research topic for large-scale similarity search owing to the low storage cost and expedited search operation. A recent research thrust aims to learn compact codes jointly from multiple sources, referred to as cross-view (or cross-modal) hashing in the literature. The main theme of this paper is to develop a novel formulation and optimization scheme for cross-view hashing. As a key differentiator, our proposed method directly conducts optimization on discrete binary hash codes, rather than relaxed continuous variables as in existing cross-view hashing methods. This way relaxation-induced search accuracy loss can be avoided. We attack the cross-view hashing problem by simultaneously capturing semantic neighboring relations and maximizing the generative probability of the learned hash codes in each view. Specifically, to enable effective optimization on discrete hash codes, the optimization proceeds in a block coordinate descent fashion. Each iteration sequentially updates a single bit with others clamped. We transform the resultant sub-problem into an equivalent, more tractable quadratic form and devise an active set based solver on the discrete codes. Rigorous theoretical analysis is provided for the convergence and local optimality condition. Comprehensive evaluations are conducted on three image benchmarks. The clearly superior experimental results faithfully prove the merits of the proposed method.

1 Introduction

Large-scale similarity search (or known as nearest neighbor search) is a critical task in computer science. Compact hashing techniques have gained much empirical success for fast similarity search in various fields. including computer vision [Torralba *et al.*, 2008; Strecha *et al.*, 2012; Liu *et al.*, 2012; Lin *et al.*, 2014], information retrieval [Chum *et al.*, 2008], data mining [Eshghi and Rajaram, 2008], machine learning [Weiss *et al.*, 2008; Salakhutdinov and Hinton, 2009; Norouzi and Fleet, 2011] and theoretic computer science [Indyk and Motwani, 1998; Charikar, 2002; Datar *et al.*, 2004]. As revealed by Johnson-Lindenstrauss lemma, under mild conditions, any n data points can be embedded into $\mathcal{O}(1/\epsilon^2)\log(n)$-dimensional space with pairwise affinity distorted at most $1 \pm \epsilon$, which sheds light on developing compact coding scheme for similarity search. The hashing methods transform original data vectors into binary Hamming space, such that the Hamming distance between two data points approximates the original pairwise dissimilarity. With the hashing techniques, each data object can be represented using only a few binary bits. The storage cost can be thereby significantly reduced, and the nearest neighbor search can be also expedited using either Hamming ranking or the data structure of hash tables.

Conventional hashing schemes mostly take single-source data as the input. Nonetheless, in the era of big data, concurrently learning compact hash codes from heterogeneous data sources is of increasing importance. For example, each YouTube video is associated with rich information in multiple heterogenous views, including a variety of visual, spatio-temporal, or audio features extracted from the video content, the descriptive text from the caption and community-contributed comments, and various video metadata (genre, time stamp, geo-tags, author etc.). One of the prominent benefits by concurrently learning hash codes from multi-view data is enabling cross-view retrieval. Specifically, one can search in some view by looking into relevant views (*e.g.*, searching image database with textual queries, or vice versa). To this end, a plurality of cross-view hashing methods have been devised. Examples can be found in [Zhen and Yeung, 2012a; 2012b; Lin *et al.*, 2015].

This paper focuses on image data and aims to elevate the performancce of hash code based cross-view image retrieval. Note that there exist two different paradigms in utilizing multi-source data for the hashing purpose: multi-view hashing and cross-view hashing. The former assumes all views are available for both the training and querying data. The latter adopts a more feasible setting, assuming only partial views are seen for the querying data. We exclude the former from the scope of this work. Our novel contribution lies in a novel cross-view hashing formulation and a corresponding optimization procedure which is characterized by the direct manipulation of discrete hash codes. Before diving into technical details we would emphasize two crucial traits of the proposed method.

First, most existing hashing methods from multi-view data still suffer from inadequate search accuracy, which can be partly imputed to the popular "relaxation + rounding" optimization strategy. To achieve high compactness, hash codes are mostly represented by binary bits. The discrete nature of hash codes makes learning the hash functions a complicated combinatorial optimization problem. Computationally, the relaxation + rounding tactic for attacking this issue is to first relax the binary variables to be continuous and afterwards round the continuous solutions. This tactic is vividly exemplified in a majority of existing cross-view hashing works such as [Kumar and Udupa, 2011; Song *et al.*, 2013; Zhang and Li, 2014; Ding *et al.*, 2014].

According to some recent empirical evaluations in [Liu *et al.*, 2014; Shen *et al.*, 2015], above relaxation + rounding schemes tend to significantly deteriorate the search accuracies, particularly when the code length becomes long. Only a few existing methods can directly conduct optimization in the hash code space. For example, ITQ [Gong and Lazebnik, 2011] learns a rotation matrix for the already-learned hash codes in order to reduce the rounding residuals. SePH [Lin *et al.*, 2015] first learns a set of "ideal" hash code purely based on the semantic supervisory information and afterwards regresses the feature vectors to these hash codes. The work in [Liu *et al.*, 2014] addresses the task of unsupervised hashing. It proposes a majorization-minimization scheme, such that each sub-problem with discrete variables can be solved in closed form and converges to some fixed point. [Shen *et al.*, 2015] is its extension in a supervised setting. Clearly, a cross-view hashing method which directly optimizes the discrete code is still missing in the literature. This paper proposes a novel method of this kind.

Second, rendering reciprocal hash bits is crucial to obtain good performance. A large body of the early development of hashing methods, such as PCA-Hash and [Wang *et al.*, 2010], require different hashing parameter vectors orthogonal to one another. This notably simplifies the numerical optimization, yet seldom entails good inter-bit complementarity with increasing code length. The works in [Xu *et al.*, 2011; Zhen and Yeung, 2012a] sequentially learn the hash bits. At a new iteration, it assigns weights to samples in an AdaBoost fashion. More emphasis is laid on the samples that most severely violate supervisory constraints using previous hash bits. Another group of works, exemplified by [Liu *et al.*, 2013], first generate a pool of over-complete hash bits, and then intelligently select most complementary subset.

Motivated by the empirical success in [Liu *et al.*, 2012], we propose an efficient block-coordinate descent optimization scheme, optimizing each hash bit conditioned on others. At each iteration, a single hash bit is chosen and others are clamped. The selected hashing function and the corresponding hash codes are greedily updated for maximally complementing others. As the prominent benefit of this scheme, each sub-problem is proved to be equivalent to some quadratic form. The resultant discrete optimization problem can be efficiently solved using active set method. A rigorous convergence proof will be also provided.

2 Discrete Cross-View Hashing

2.1 Notations

Throughout this paper we use boldface type for vectors, matrices or their collections. Assume we have a training set $\mathcal{X}$ that comprises of n data objects. Each data object is described by V distinct views. Let $\mathbf{x}_i = (\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(V)}) \in \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_V}$ be the feature collection for data object i, where $\mathbf{x}_i^{(v)}$ denotes the feature extracted from the v-th view. Following the majority of supervised hashing algorithms, let us assume that the supervision information is provided in a pairwise style. In particular, let $\mathcal{S}$, $\mathcal{D}$ collect all similar / dissimilar pairs respectively. For notational convenience, we further introduce a supervision matrix $\mathbf{Y} \in \{-1, 0, 1\}^{n \times n}$ as

$$\mathbf{Y}_{i,j} = \begin{cases} 1, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\ -1, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Let K be the desired number of hash bits and $h_k : \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_V} \mapsto \{-1, 1\}, k = 1, \ldots, K$ be the corresponding hashing functions. For notational conciseness, let us stack the hash bits for all data objects, obtaining a code matrix $\mathbf{B} \in \{-1, 1\}^{n \times K}$ as below:

$$\mathbf{B} = \begin{bmatrix} b_1(\mathbf{x}_1), & b_2(\mathbf{x}_1), & \ldots, & b_K(\mathbf{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(\mathbf{x}_n), & b_2(\mathbf{x}_n), & \ldots, & b_K(\mathbf{x}_n) \end{bmatrix}.$$

For brevity, denote the k-th column as $\mathbf{B}_k$ and the other $k-1$ columns as $\mathbf{B}_{\setminus k}$. For data object $\mathbf{x}$, we use the notation $\mathbf{b}(\mathbf{x})$ to encapsulate the hash bits $b_1(\mathbf{x}), \ldots, b_K(\mathbf{x})$, and $\mathbf{b}_{\setminus k}(\mathbf{x})$ for all of its hash bits except for the k-th one.

2.2 Objective Design

The proposed objective combines two considerations:

Criterion-I: semantic preservation. In the context of supervised hashing, it shall be critically ensured that similar samples have alike hash codes. We first adopt the notion of *code product*, which admits a one-to-one correspondence to Hamming distance and relatively easier for manipulation. A normalized version of code product ranging over $[-1, 1]$ is described as

$$\mathbf{b}(\mathbf{x}_i) \circ \mathbf{b}(\mathbf{x}_j) = \tfrac{1}{K} \textstyle\sum_{k=1}^{K} b_k(\mathbf{x}_i) b_k(\mathbf{x}_j). \tag{2}$$

For each data pair $(\mathbf{x}_i, \mathbf{x}_j)$, we would define a merit function $\ell(\mathbf{x}_i, \mathbf{x}_j)$. Intuitively, the merit function should faithfully reflect the semantic consistency between the code product and corresponding indicator $\mathbf{Y}_{i,j}$. Our proposed formulation supports a large spectrum of merit functions. $\ell(\mathbf{x}_i, \mathbf{x}_j)$ is assumed to be linear with respect to $\mathbf{b}(\mathbf{x}_i) \circ \mathbf{b}(\mathbf{x}_j)$ in order to ensure algorithmic convergence. Below are two examples of valid merit functions:

$$\begin{aligned} (\mathit{exponential}): &\quad \ell(\mathbf{x}_i, \mathbf{x}_j) = e^{\mathbf{Y}_{i,j} \cdot \mathbf{b}(\mathbf{x}_i) \circ \mathbf{b}(\mathbf{x}_j)}. \\ (\mathit{linear}): &\quad \ell(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{Y}_{i,j} \cdot \mathbf{b}(\mathbf{x}_i) \circ \mathbf{b}(\mathbf{x}_j). \end{aligned}$$

Criterion-II: logistic generative probability. We also propose to leverage a generative model that estimates the probability of mapping a view-specific feature vector to some

hash bit value. The purposes are two-folds: first, optimizing $\ell(\mathbf{x}_i, \mathbf{x}_j)$ by ignoring the feature vectors may bring some hash code matrix $\mathbf{B}$ which perfectly satisfies all semantic supervisory constraints. We here argue that it is also important to ascertain such perfect hash codes are practically feasible based on view-specific feature vectors. And secondly, when tackling unseen querying data objects, the generative model returns their most likely hash bits.

For simplicity, we adopt linear logistic regression as the probabilistic building block, namely

$$\phi_k^{(v)}(b, \mathbf{x}_i^{(v)}) = 1/(1 + e^{-b \cdot \mathbf{w}_{k,v}^\top \mathbf{x}_i^{(v)}}), \tag{3}$$

where the superscript and subscript of $\phi(\cdot, \cdot)$ collectively emphasize that the probability is for the k-th hash bit from v-th data view. $\mathbf{w}_{k,v}$ denotes the parameter vector whose length can be inferred from the context. Clearly, larger values of ϕ are always favored since they imply that the learned bit $\mathbf{b}$ has better chance to be alternatively inferred from the features.

By compiling all training data, we obtain the overall maximization problem below:

$$\max_{\mathbf{B}, \{\mathbf{w}_{k,v}\}} \frac{1}{n^2} \sum_{i,j=1}^{n} \ell(\mathbf{x}_i, \mathbf{x}_j) + \frac{\gamma}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{v=1}^{V} \phi_k^{(v)}(b_k(\mathbf{x}_i), \mathbf{x}_i^{(v)}),$$

where $\gamma > 0$ is a free parameter.

3 Alternating Maximization

The proposed cross-view hashing formulation is essentially a mixed-integer optimization problem with respect to both the discrete hash code $\mathbf{B}$ and continuous variables $\{\mathbf{w}_{k,v}\}$. The coupling of $\mathbf{B}$ and $\{\mathbf{w}_{k,v}\}$ in Eqn. (3) significantly complicates the pursuit of the optimal solutions. In fact, setting $\gamma = 0$ and adopting the linear metric function, the objective boils down to the Max-Cut problem and no polynomial-time algorithm is known to exist for achieving the global maximum. We thus adopt an alternating maximization strategy for the consideration of numerical tractability. Namely, we iterate between updating $\mathbf{B}$ with $\{\mathbf{w}_{k,v}\}$ frozen ($\mathbf{B}$-Subproblem) or vice verse ($\mathbf{W}$-Subproblem), which are detailed in subsequent sections respectively.

3.1 B-Subproblem

Without loss of generality, here we instantiate the merit function using the non-linear exponential form. The algorithmic description and analysis in this section also apply for other merit functions if they are linear w.r.t. the code product. Knowing $\{\mathbf{w}_{k,v}\}$, the $\mathbf{B}$-subproblem is derived as

$$\max_{\mathbf{B}} \frac{1}{n^2} \sum_{i,j} e^{\mathbf{Y}_{i,j} \cdot \mathbf{b}(\mathbf{x}_i) \circ \mathbf{b}(\mathbf{x}_j)} + \frac{\gamma}{n} \sum_{i,k,v} \phi_k^{(v)}(b_k(\mathbf{x}_i), \mathbf{x}_i^{(v)}). \tag{4}$$

We tackle the above subproblem by sequentially updating the hash bits in K steps. Step k conducts a block coordinate ascent update on the k-th column of code matrix $\mathbf{B}$, namely $\mathbf{B}_k$. Hereafter we illustratively describe the procedure for updating bit k.

Coordinate discrete quadratic program: Importantly, we can derive an equivalent quadratic form for each sub-problem

Algorithm 1 Coordinate Discrete Optimization Procedure

Input: multi-view data $\mathcal{X}$, indicator matrix $\mathbf{Y}$, parameter γ.
Output: $\mathbf{B}$, $\{\mathbf{w}_{k,v}\}$.
Main Loop
Initialize $\{\mathbf{w}_{k,v}\}$ using Gaussian random numbers and initialize $\mathbf{B}$ according to $b_k(\mathbf{x}_i) = \mathbf{sign}(\sum_{v=1}^{V} \mathbf{w}_{k,v}^\top \mathbf{x}_i^{(v)})$;
while not converged **do**
  Solve $\mathbf{B}$-subproblem for updating the code matrix $\mathbf{B}$;
  Solve $\mathbf{W}$-subproblem for updating parameter $\{\mathbf{w}_{k,v}\}$;
end while
B-subproblem
for $k = 1$ **to** K **do**
  Calculate matrices $\mathbf{A}$ and $\mathbf{Z}$ from other $K-1$ bits $\mathbf{B}_{\setminus k}$, according to Eqns. (6)(7) respectively.
  Calculate $\widehat{\mathbf{B}}_k$ by Eqn. (11) and obtain active set $\mathcal{I}$;
  Conduct active set reduction with Algorithm 2;
  Update $\mathbf{B}_k$ according to Eqn. (13);
end for
W-subproblem
for $(k, v) \in \{1 \dots K\} \times \{1 \dots V\}$ **do**
  Use stochastic gradient ascent to solve Problem (14) and update $\mathbf{w}_{k,v}$ accordingly;
end for

defined on a single hash bit, which admits an efficient discrete solution. Let us first define the leave-one-out partial code product as $\mathbf{b}_{\setminus k}(\mathbf{x}_i) \circ \mathbf{b}_{\setminus k}(\mathbf{x}_j) = \mathbf{b}(\mathbf{x}_i) \circ \mathbf{b}(\mathbf{x}_j) - \frac{1}{K} b_k(\mathbf{x}_i) b_k(x_j)$. For the exponential merit function, it is easily verified

$$\begin{aligned} & e^{\mathbf{Y}_{i,j}(\mathbf{b}(\mathbf{x}_i) \circ \mathbf{b}(\mathbf{x}_j))} \\ = \; & e^{\mathbf{Y}_{i,j}\left(\mathbf{b}_{\setminus k}(\mathbf{x}_i) \circ \mathbf{b}_{\setminus k}(\mathbf{x}_j)\right)} \cdot e^{\frac{1}{K} \mathbf{Y}_{i,j} \cdot b_k(\mathbf{x}_i) b_k(x_j)}, \\ = \; & e^{\mathbf{Y}_{i,j}\left(\mathbf{b}_{\setminus k}(\mathbf{x}_i) \circ \mathbf{b}_{\setminus k}(\mathbf{x}_j)\right)} \cdot \left[c_{i,j} + c'_{i,j} \cdot b_k(\mathbf{x}_i) b_k(x_j)\right], \end{aligned}$$

where $c_{i,j} = \frac{1}{2}(e^{\frac{1}{K}\mathbf{Y}_{i,j}} + e^{-\frac{1}{K}\mathbf{Y}_{i,j}})$ and $c'_{i,j} = \frac{1}{2}(e^{\frac{1}{K}\mathbf{Y}_{i,j}} - e^{-\frac{1}{K}\mathbf{Y}_{i,j}})$ are sample-specific constants. The last equality is obtained based on a simple observation: although $\ell(\mathbf{x}_i, \mathbf{x}_j)$ is discrete and non-linear, $b_k(\mathbf{x}_i) b_k(\mathbf{x}_j)$ can only take values ± 1. Therefore, we can bi-linearize $e^{-\frac{1}{K}\mathbf{Y}_{i,j} \cdot b_k(\mathbf{x}_i) b_k(\mathbf{x}_j)}$ through exhaustively enumerating all possible values.

Using the same idea, we can also linearize $\phi_k^{(v)}(b, \mathbf{x}_i^{(v)})$ with respect to $b_k(\mathbf{x}_i)$, obtaining

$$\phi_k^{(v)}(b, \mathbf{x}_i^{(v)}) = d_{i,k,v} + d'_{i,k,v} \cdot b_k(\mathbf{x}_i), \tag{5}$$

where $d_{i,k,v} = 1/2$ and $d'_{i,k,v} = 1/(1 + e^{-\mathbf{w}_{k,v}^\top \mathbf{x}_i^{(v)}}) - 1/2$.

For simplifying the notations, let us introduce two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$, whose (i, j)-th elements are

$$\mathbf{A}_{i,j} = e^{\mathbf{Y}_{i,j}\left(\mathbf{b}_{\setminus k}(\mathbf{x}_i) \circ \mathbf{b}_{\setminus k}(\mathbf{x}_j)\right)} \cdot c'_{i,j}, \tag{6}$$

$$\mathbf{Z}_{i,j} = \sum_{v=1}^{V} d'_{i,k,v}. \tag{7}$$

Updating bit k can be accomplished by solving the following quadratic polynomial involving binary discrete vector $\mathbf{B}_k$, whose equivalence to Problem (4) naturally holds due to above linearization transforms:

$$\max_{\mathbf{B}_k \in \{\pm 1\}^n} \mathcal{J}(\mathbf{B}_k) \triangleq \frac{1}{n^2} \mathbf{B}_k^\top \mathbf{A} \mathbf{B}_k + \frac{1}{n} \gamma \mathbf{Z} \mathbf{B}_k + const, \tag{8}$$

Algorithm 2 Active Set Reduction

Input: active set $\mathcal{I}$, matrices $\mathbf{A}, \mathbf{Z}$, hash vector $\mathbf{B}_k$.
Output: a reduced active set $\mathcal{I}_r \subseteq \mathcal{I}$.
Set indicator $\mathbf{s} \in \{0,1\}^n$ as 1 for all entries in $\mathcal{I}$, otherwise 0;
Calculate $\mathbf{M} = \mathbf{A} \odot (\mathbf{B}_k \mathbf{B}_k^\top) \odot (\mathbf{s}\mathbf{s}^\top)$;
Initialize $\mathcal{I}_r = \{i\}$, where $i \in \mathcal{I}$ is randomly chosen;
$\mathbf{v}_{gain} \leftarrow \mathbf{M}(i,:)$, where $\mathbf{M}(i,:)$ denotes row i of $\mathbf{M}$;
while true **do**
  Find the largest one from the set $\mathcal{I} \setminus \mathcal{I}_r$:

$$i = \arg_{i \in \mathcal{I} \setminus \mathcal{I}_r} \max \mathbf{v}_{gain}(i). \quad (9)$$

$$q = \max_{i \in \mathcal{I} \setminus \mathcal{I}_r} \mathbf{v}_{gain}(i). \quad (10)$$

  if $q < 0$ **do**, **break**; **end if**
  Expand the target index set $\mathcal{I}_r \leftarrow \mathcal{I}_r \cup \{i\}$;
  Update the indicator vector $\mathbf{v}_{gain} \leftarrow \mathbf{v}_{gain} + \mathbf{M}(i,:)$, reflecting the (scaled) gain of adding the next element into $\mathcal{I}_r$;
end while

Active set construction and reduction: Optimizing Problem (8) can be intuitively understood as flipping some hash bits so as to elevate the objective value. Noting that most samples are prone to their original hash bit values, we construct an "active set" with few samples which have the maximal potentials of being flipped. This can be efficiently completed by a signed-gradient scheme:

$$\widehat{\mathbf{B}}_k \leftarrow \mathbf{sign}\left(\frac{1}{n^2} 2\mathbf{A}\mathbf{B}_k + \frac{1}{n}\gamma\mathbf{Z}\right), \quad (11)$$

and the active set is set as $\mathcal{I} = \{i \mid \widehat{\mathbf{B}}_k(i) \neq \mathbf{B}_k(i)\}$.

However, simultaneously flipping all in the active set does not ensure increasing the objective. Let $\mathbf{B}'_k$ be the final new binary solution. We later prove that a sufficient condition for a better solution is the non-negativity of the quantity below:

$$(\mathbf{B}'_k - \mathbf{B}_k)^\top \mathbf{A} (\mathbf{B}'_k - \mathbf{B}_k) \geq 0. \quad (12)$$

To this end, we propose an efficient active set screening procedure as described in Algorithm 2, which returns a subset of $\mathcal{I}$ that rigorously satisfies Condition (12). The idea of this procedure is initializing $\widehat{\mathbf{B}}_k = \mathbf{B}_k$ and greedily flipping one more bit from the original active set $\mathcal{I}$ at each step. The vector $\mathbf{v}_{gain}$ therein denotes the increase of $(\mathbf{B}'_k - \mathbf{B}_k)^\top \mathbf{A} (\mathbf{B}'_k - \mathbf{B}_k)$ by flipping corresponding bit. The process terminates until the quantity in Eqn. (12) stops increasing (*i.e.*, $q < 0$ in Algorithm 2). The new binary solution is then calculated as

$$\mathbf{B}'_k(i) = \begin{cases} \mathbf{B}_k(i), & i \notin \mathcal{I}_r \\ -\mathbf{B}_k(i), & i \in \mathcal{I}_r \end{cases} \quad (13)$$

Regarding the convergence analysis, we have the major theoretic observations below:

Theorem 1. $\mathcal{J}(\mathbf{B}'_k) \geq \mathcal{J}(\mathbf{B}_k)$.

Proof. Let us define an auxiliary function $g(\mathbf{X}) = \left(2\mathbf{A}\mathbf{B}_k/n^2 + \gamma\mathbf{Z}/n\right)^\top (\mathbf{X} - \mathbf{B}_k)$, where $\mathbf{X} \in \{\pm 1\}^n$. Note that $g(\mathbf{B}_k) = 0$ and in Eqn. (11), $\widehat{\mathbf{B}}_k$ is indeed a unique maximizer of the problem $\widehat{\mathbf{B}}_k = \arg_{\mathbf{X} \in \{\pm 1\}^n} \max g(\mathbf{X})$. Therefore $g(\widehat{\mathbf{B}}_k) \geq g(\mathbf{B}_k) = 0$. Recall that $\widehat{\mathbf{B}}_k$ induces an active set $\mathcal{I}$. Since $g(\mathbf{X})$ is separable w.r.t. each binary variable in $\mathbf{X}$, a solution induced by any subset of $\mathcal{I}$ will also increase the value of $g(\cdot)$ compared with $g(\mathbf{B}_k)$. Hence the solution $\mathbf{B}'_k$ calculated according to Eqn. (13) represents some better solution compared to $\mathbf{B}_k$, namely $g(\mathbf{B}'_k) \geq g(\mathbf{B}_k) = 0$. It is verified that

$$\mathcal{J}(\mathbf{B}'_k) - \mathcal{J}(\mathbf{B}_k) = g(\mathbf{B}'_k) + \left(\mathbf{B}'_k - \mathbf{B}_k\right)^\top \mathbf{A} \left(\mathbf{B}'_k - \mathbf{B}_k\right).$$

By the constructions in (12) and (13), both terms on the right hand side are non-negative, which proves the claim. □

$\mathbf{B}'_k$ will eventually converge to some fixed point, as described below.

Theorem 2. $\mathbf{B}_k$ *converges to a fixed point that satisfies* $\mathbf{sign}\left(2\mathbf{A}\mathbf{B}_k/n^2 + \gamma\mathbf{Z}/n\right) = \mathbf{B}_k$*, in the sense that flipping one more bit of* $\mathbf{B}_k$ *will not further increase the objective value of Problem (8).*

Proof. The claim can be established through proof of contradiction. Flipping any bit of $\mathbf{B}_k$ incurs some change of the objective value of Problem (8). Note that $\mathbf{A}$ is a symmetric matrix and its diagonal elements are all zeros. It can be verified that the change by flipping a bit is exactly the corresponding quantity in $\left(2\mathbf{A}\mathbf{B}_k/n^2 + \gamma\mathbf{Z}/n\right) \odot (-\mathbf{B}_k)$, where $\odot$ denotes point-wise multiplication. Once all entries of $\left(2\mathbf{A}\mathbf{B}_k/n^2 + \gamma\mathbf{Z}/n\right) \odot (-\mathbf{B}_k)$ become non-positive (namely $\mathbf{sign}\left(2\mathbf{A}\mathbf{B}_k/n^2 + \gamma\mathbf{Z}/n\right) = \mathbf{B}_k$), $\mathcal{J}(\mathbf{B}_k)$ can not be further improved by flipping another bit. Otherwise, we can flip any bit with positive correspondence in $\left(2\mathbf{A}\mathbf{B}_k/n^2 + \gamma\mathbf{Z}/n\right) \odot (-\mathbf{B}_k)$ to further increase $\mathcal{J}(\mathbf{B}'_k)$. □

Remark on complexity: The complexity of $\mathbf{B}$-subproblem mainly comes from matrices $\mathbf{A}$, $\mathbf{Z}$, vector $\widehat{\mathbf{B}}_k$, and active set reduction. It is easily verified that the complexities of computing $\mathbf{A}, \mathbf{Z}, \widehat{\mathbf{B}}_k$ are $\mathcal{O}(n^2), \mathcal{O}(Vn^2), \mathcal{O}(n^2)$ respectively. In Algorithm 2, iteratively updating $\mathbf{v}_{gain}$ can be in an $\mathcal{O}(n^2)$ complexity in the worst case (*i.e.*, a majority of samples are included in the initial active set and most are chosen into $\mathcal{I}_r$ by Algorithm 2). In practice, the size of active set quickly shrinks when the optimization proceeds. In addition, we also adopt a data chunk based trick for addressing large n, as detailed in the experimental section.

3.2 W-Subproblem

For each bit-view pair $(k, v) \in \{1 \dots K\} \times \{1 \dots V\}$, there exists an associated parameter $\mathbf{w}_{k,v}$. When the code matrix $\mathbf{B}$ is fixed, they can be updated by separately solving $K \times V$ sub-problems. Each is defined as below:

$$\mathbf{w}_{k,v} \leftarrow \arg_{\mathbf{w}_{k,v}} \max \frac{\gamma}{n} \sum_{i=1}^{n} \phi_k^{(v)}(b_k(\mathbf{x}_i), \mathbf{x}_i^{(v)}). \quad (14)$$

Recall that we adopt a logistic form for all $\phi_k^{(v)}(\cdot,\cdot)$, which is amenable for stochastic gradient ascend (SGA). Specifically, the stochastic gradient for data object $\mathbf{x}_i$ is computed by

$$\begin{aligned} & \partial \phi_k^{(v)}(b_k(\mathbf{x}_i), \mathbf{x}_i^{(v)}) / \partial \mathbf{w}_{k,v} \\ = \; & \left[\phi_k^{(v)}(b_k(\mathbf{x}_i), \mathbf{x}_i^{(v)})(1 - \phi_k^{(v)}(b_k(\mathbf{x}_i), \mathbf{x}_i^{(v)}))b_k(\mathbf{x}_i)\right] \cdot \mathbf{x}_i^{(v)}. \end{aligned}$$

	Method	Wiki			NUS-WIDE			MIRFlickr		
		16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Image Query	CVH	0.1257	0.1212	0.1215	0.3687	0.4182	0.4602	0.6067	0.6177	0.6157
	IMH	0.1573	0.1575	0.1568	0.4187	0.3975	0.3778	0.6016	0.6120	0.6070
	CMSSH	0.1877	0.1771	0.1646	0.4063	0.3927	0.3939	0.5728	0.5743	0.5706
	CHMIS	0.2107	0.2089	0.2026	0.4670	0.4696	0.4635	0.6027	0.6002	0.5947
	LSSH	0.2141	0.2216	0.2218	0.3900	0.3924	0.3962	0.5784	0.5804	0.5797
	SCM	0.2210	0.2337	0.2442	0.4842	0.4941	0.4947	0.6237	0.6343	0.6448
	CMFH	0.2132	0.2259	0.2362	0.4267	0.4229	0.4207	0.5861	0.5835	0.5844
	SePH-RBF	0.2762	0.2965	0.3049	0.5394	0.5454	0.5499	0.6720	0.6761	0.6749
	SePH-lin	0.1842	0.2016	0.2132	0.5306	0.5382	0.5439	0.6620	0.6654	0.6697
	Our CDH-0	0.3116	0.3493	0.3611	**0.5846**	**0.5898**	0.5995	0.6819	**0.7170**	0.7204
	Our CDH	**0.3289**	**0.3608**	**0.3615**	0.5768	**0.5898**	**0.6050**	**0.7032**	**0.7170**	**0.7324**
Text Query	CVH	0.1185	0.1034	0.1024	0.3646	0.4024	0.4339	0.6026	0.6041	0.6017
	IMH	0.1463	0.1311	0.1290	0.4053	0.3892	0.3758	0.5895	0.6031	0.6010
	CMSSH	0.1630	0.1617	0.1539	0.3874	0.3849	0.3704	0.5715	0.5732	0.5699
	CHMIS	0.1807	0.1789	0.1726	0.4470	0.4496	0.4435	0.5827	0.5802	0.5747
	LSSH	0.5031	0.5224	0.5293	0.4286	0.4248	0.4208	0.5898	0.5927	0.5932
	SCM	0.2134	0.2366	0.2479	0.4536	0.4620	0.4630	0.6133	0.6209	0.6295
	CMFH	0.4884	0.5132	0.5269	0.4627	0.4556	0.4518	0.5937	0.5919	0.5931
	SePH-RBF	0.6312	0.6581	0.6637	0.6230	0.6331	0.6407	0.7178	0.7243	0.7287
	SePH-lin	0.5428	0.5665	0.5724	0.6220	0.6342	0.6459	0.7086	0.7163	0.7232
	Our CDH-0	0.7121	0.7219	0.7239	0.6757	**0.7011**	0.7061	**0.7702**	**0.7980**	0.7903
	Our CDH	**0.7400**	**0.7309**	**0.7380**	**0.6866**	**0.7011**	**0.7171**	0.7596	**0.7980**	**0.7941**

Table 1: Cross-view image retrieval performance with varying code lengths in terms of mAP scores. SePH-RBF and SePH-lin correspond to the variants of SePH using RBF-kernelized or line hash functions respectively. CDH denotes our proposed algorithm. CDH-0 is the variant by setting the parameter γ to 0. The best performers are highlighted in boldface.

With the gradient calculation formula, the use of SGA in Problem (14) is quite standard. We thus omit more details.

3.3 Generative Model for Out-of-Sample Data

We have introduced a generative model for each view, whose posterior is defined in Eqn. (3). These generative models can faithfully infer the most possible hash code for an unseen data object, either with all views or partial views available. Let us first consider the all-view case. Given a data object $\mathbf{x}$, we estimate its k-th bit by maximizing the joint probability $b_k = \arg_{b\in\{\pm 1\}} \max\ P_k(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(V)}, b)$.

Assume different views are independent to each other conditioned on b. Applying Bayes' rules gets

$$\begin{aligned} P_k(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(V)}, b) &= P_k(b) \textstyle\prod_{v=1}^{V} P_k(\mathbf{x}^{(v)}|b) \\ &= P_k(b) \prod_{v=1}^{V} \frac{P_k(b|\mathbf{x}^{(v)}) P_k(\mathbf{x}^{(v)})}{P_k(b)} \\ &\propto (P_k(b))^{1-V} \textstyle\prod_{v=1}^{V} \phi_k^{(v)}(b, \mathbf{x}^{(v)}), \end{aligned}$$

where $P_k(b)$ represents the prior of value b and can be empirically set as its percentage in the learned codes $\mathbf{B}_k$ on the training data.

The joint probability for partial views can be derived in a similar spirit. In cross-view hashing, typically only one view (*e.g.*, textual or visual view for image data) is seen for a querying data object. In this extreme case, we have $P_k(\mathbf{x}^{(v)}, \pm 1) = P_k(\pm 1)\phi_k^{(v)}(\pm 1, \mathbf{x}^{(v)})$ when only view v is observed. Its hash bit can be inferred by simply comparing $P_k(\mathbf{x}^{(v)}, 1)$ and $P_k(\mathbf{x}^{(v)}, -1)$.

4 Experiment

This section reports the evaluation results of the proposed cross-view hashing method and state-of-the-art competitors. Hereafter we denote the proposed method as CDH (**C**oordinate **D**iscrete **H**ashing).

Datasets: We adopt three datasets, including Wiki [Rasiwasia *et al.*, 2010], NUS-WIDE [seng Chua *et al.*, 2009] and MIRFlickr [Huiskes and Lew, 2008]. For all three, our data preparation is essentially identical to other relevant works.

The Wiki dataset is crawled from Wikipedia's featured documents. It consists of 2,866 image-text pairs, which are annotated with semantic labels of 10 classes. We represent image with 128-dimensional SIFT visual feature and 10-dimensional LDA topical feature. 75% of the data are used for learning the hash functions and other 25% are queries.

The NUS-WIDE dataset contains 269,648 user-generated images from Flickr.com. The experiment chooses 10 most common tags and 186,577 images associated by at least one of these tags. Each image is represented by 500-D SIFT features and 1,000-D binary indicator vector for 1,000 most frequent tags appearing in NUS-WIDE. 99% and 1% of the data are for training and querying respectively.

The MIRFlickr dataset consists of 25,000 image-text pairs from 24 unique categories. For each image object, 150-D edge histogram and 500-D textual feature are extracted. The ratios of data for train/query are 95% and 5% respectively.

Baselines and Evaluation Protocol: Our method is contrasted with a large spectrum of representative cross-view hashing methods, including *CVH* [Kumar and Udupa, 2011], *IMH* [Song *et al.*, 2013], *CMSSH* [Bronstein *et al.*, 2010],

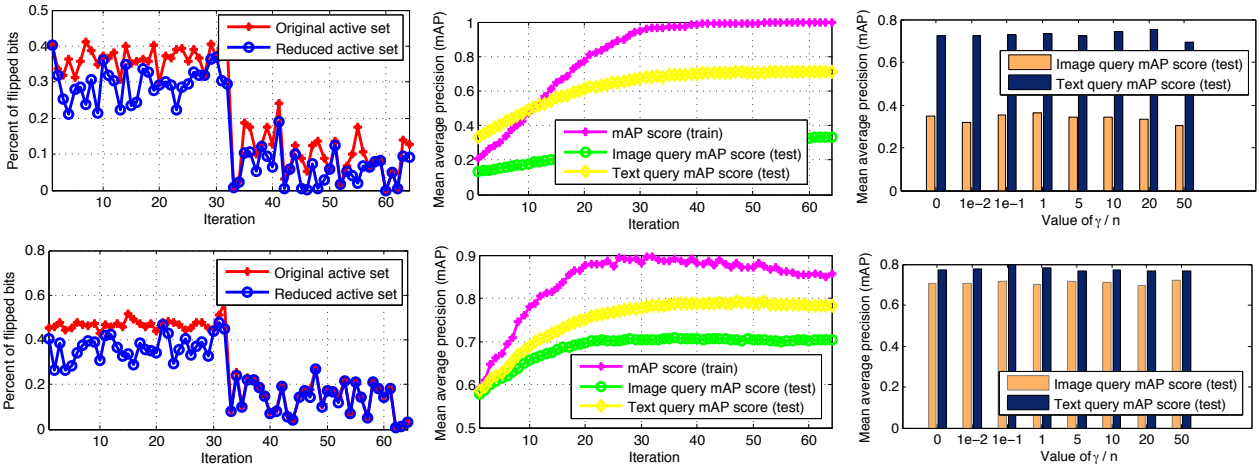

Figure 1: *Left column*: active set sizes beore/after the reduction in Algorithm 2. *Middle column*: image search mAP scores during the training and querying (testing) stages. *Right column*: sensitivity study w.r.t. parameter γ/n. The two rows are results for Wiki and MIRFlickr obtained with 32 hash bits respectively. $\gamma/n = 1$ for obtaining the results in left two columns.

CHMIS [Zhang *et al.*, 2011], *LSSH* [Zhou *et al.*, 2014], *SCM* [Zhang and Li, 2014], *CMFH* [Ding *et al.*, 2014], *SePH* [Lin *et al.*, 2015]. We thank all authors sharing the core source codes of the algorithms. The parameters are adopted either as the defaulted ones in the software package or the suggested values in the original papers. Following the evaluation protocol of most prior works, we adopt a Hamming ranking framework for evaluating the cross-view hashing performance. The querying data objects are assumed to have partial views (visual or textual view) seen. The hash code for the queries are calculated by the probabilistic inference model as described in Section 3.3. All database samples are ranked based on the Hamming distances to the query, and a *mean average precision* (mAP) score is calculated from the ranked lists as the primary indicator of the search performance.

The parameter γ in our proposed CDH plays a role of balancing the learned hash code quality and the generalization ability to unseen data. We use a grid search scheme to find the optimal γ on all datasets. Particularly, multiple trials are conducted with different γ/n values from the candidate set $[0, 10^{-2}, 10^{-1}, 1, 5, 10, 20, 50]$. The value with the highest objective value on the training data is found and the corresponding accuracies are reported. We initialize the hashing parameters $\mathbf{w}_{k,v}$ using random numbers drawn from 1-D Gaussian distribution, and afterwards initialize the hash bit by $b_k(\mathbf{x}) = \text{sign}[\sum_{v=1}^{V}(\phi_k^{(v)}(1, \mathbf{x}^{(v)}) - 0.5)]$. The largest dataset NUSWIDE contains more than 180,000 images. Since the complexity of Algorithm 2 is quadratic w.r.t. sample count, directly manipulating the entire dataset requires tremendous memory. We thus randomly split the data into several chunks, each of which contains 10,000 images. The active set scheme is separately conducted on each data chunk.

Investigation of Experimental Results: The mAP scores for all methods are reported in Table 1. It is obvious that our proposed CDH significantly outperforms all baseline methods by 5-20% absolute accuracy gain on all benchmarks with different hash code lengths. We attribute this empirical superiority to the discrete optimization scheme. Moreover, to emphasize the generative probability term in the objective, we include a variant CDH-0, which sets $\gamma = 0$ to rule out the generative models. The comparisons reveal that on most tasks a positive γ brings notable improvement.

Figure 1 presents another three sets of results obtained on Wiki and MIRFlickr. The leftmost column displays the percentage of flipped bits (proportional to the sizes of active sets $\mathcal{I}$ and $\mathcal{I}_r$ in Algorithm 2) at different optimization iterations. Clearly, both sets quickly shrink as the optimization proceeds. The middle column shows the evolution of the search accuracy of cross-view querying. Since the object values during training are not intuitively understood, we estimate another search accuracy at the training stage by randomly choosing 1,000 training samples as the queries. We find the optimization converges very fast. In practice, after scanning all K bits two passes, both the objective values and cross-view retrieval accuracy become stable. Therefore all experiments are set to terminate after updating the hash bits at most two passes. The rightmost column investigates the mAP scores under different choices of the parameter γ. It is observed that, although each dataset favors specific γ to achieve its peak performance, the accuracies are stable with a large range of γ values.

5 Concluding Remarks

We have presented a novel formulation for hash code based cross-view image retrieval. The defining traits include an optimization scheme which directly manipulate the discrete binary variables, and a coordinate ascend method which brings a simplified sub-problem in quadratic form. We provide rigorous theoretic analysis about the convergence and local optimality condition. Moreover, our quantitative evaluations on three benchmarks also strongly validate its superiority.

Acknowledgement: Part of the research work was conducted when the first author worked at the Multimedia De-

partment of AT&T Labs, U.S.A.. This work is supported by a start-up research grant provided by Institute of Computer Science and Technology at Peking University, China.

References

[Bronstein *et al.*, 2010] Michael M. Bronstein, Alexander M. Bronstein, Fabrice Michel, and Nikos Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*, 2010.

[Charikar, 2002] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[Chum *et al.*, 2008] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, 2008.

[Datar *et al.*, 2004] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *STOC*, 2004.

[Ding *et al.*, 2014] Guiguang Ding, Yuchen Guo, and Jile Zhou. Collective matrix factorization hashing for multimodal data. In *CVPR*, 2014.

[Eshghi and Rajaram, 2008] Kave Eshghi and Shyamsundar Rajaram. Locality sensitive hash functions based on concomitant rank order statistics. In *ACM SIGKDD*, 2008.

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

[Huiskes and Lew, 2008] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *MIR '08*, 2008.

[Indyk and Motwani, 1998] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, 1998.

[Kumar and Udupa, 2011] Shaishav Kumar and Raghavendra Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, 2011.

[Lin *et al.*, 2014] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 2014.

[Lin *et al.*, 2015] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. Semantics-preserving hashing for cross-view retrieval. In *CVPR*, 2015.

[Liu *et al.*, 2012] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, 2012.

[Liu *et al.*, 2013] Xianglong Liu, Junfeng He, and Bo Lang. Reciprocal hash tables for nearest neighbor search. In *AAAI*, 2013.

[Liu *et al.*, 2014] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih fu Chang. Discrete graph hashing. In *NIPS*. 2014.

[Norouzi and Fleet, 2011] Mohammad Norouzi and David J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[Rasiwasia *et al.*, 2010] Nikhil Rasiwasia, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert R.G. Lanckriet, Roger Levy, and Nuno Vasconcelos. A new approach to cross-modal multimedia retrieval. In *ACM Multimedia*, 2010.

[Salakhutdinov and Hinton, 2009] Ruslan Salakhutdinov and Geoffrey E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.

[seng Chua *et al.*, 2009] Tat seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE: A real-world web image database from national university of singapore. In *CIVR*, 2009.

[Shen *et al.*, 2015] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *CVPR*, 2015.

[Song *et al.*, 2013] Jingkuan Song, Yang Yang, Yi Yang, Zi Huang, and Heng Tao Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *SIGMOD*, 2013.

[Strecha *et al.*, 2012] C. Strecha, A. M. Bronstein, M. M. Bronstein, and Pascal Fua. LDAHash: improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1), 2012.

[Torralba *et al.*, 2008] Antonio Torralba, Robert Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[Wang *et al.*, 2010] Jun Wang, S. Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

[Weiss *et al.*, 2008] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, 2008.

[Xu *et al.*, 2011] Hao Xu, Jingdong Wang, Zhu Li, Gang Zeng, Shipeng Li, and Nenghai Yu. Complementary hashing for approximate nearest neighbor search. In *ICCV*, 2011.

[Zhang and Li, 2014] Dongqing Zhang and Wu-Jun Li. Large-scale supervised multimodal hashing with semantic correlation maximization. In *AAAI*, 2014.

[Zhang *et al.*, 2011] Dan Zhang, Fei Wang, and Luo Si. Composite hashing with multiple information sources. In *ACM SIGIR*, 2011.

[Zhen and Yeung, 2012a] Yi Zhen and Dit-Yan Yeung. Co-regularized hashing for multimodal data. In *NIPS*, 2012.

[Zhen and Yeung, 2012b] Yi Zhen and Dit-Yan Yeung. A probabilistic model for multimodal hash function learning. In *SIGKDD*, 2012.

[Zhou *et al.*, 2014] Jile Zhou, Guiguang Ding, and Yuchen Guo. Latent semantic sparse hashing for cross-modal similarity search. In *SIGIR '14*, 2014.