Learning Binary Codes for Maximum Inner Product Search

Fumin Shen[†], Wei Liu[♯], Shaoting Zhang[‡], Yang Yang[†] and Heng Tao Shen[†§] *
[†] University of Electronic Science and Technology of China, [♯] Xidian University
[‡] University of North Carolina at Charlotte, [§] The University of Queensland

Abstract

Binary coding or hashing techniques are recognized to accomplish efficient near neighbor search, and have thus attracted broad interests in the recent vision and learning studies. However, such studies have rarely been dedicated to Maximum Inner Product Search (MIPS), which plays a critical role in various vision applications. In this paper, we investigate learning binary codes to exclusively handle the MIPS problem. Inspired by the latest advance in asymmetric hashing schemes, we propose an asymmetric binary code learning framework based on inner product fitting. Specifically, two sets of coding functions are learned such that the inner products between their generated binary codes can reveal the inner products between original data vectors. We also propose an alternative simpler objective which maximizes the correlations between the inner products of the produced binary codes and raw data vectors. In both objectives, the binary codes and coding functions are simultaneously learned without continuous relaxations, which is the key to achieving high-quality binary codes. We evaluate the proposed method, dubbed Asymmetric Inner-product Binary Coding (AIBC), relying on the two objectives on several large-scale image datasets. Both of them are superior to the state-of-the-art binary coding and hashing methods in performing MIPS tasks.

1. Introduction

In the recent years, binary coding (also known as hashing) has become a very popular research subject in computer vision [19, 22, 33], machine learning [16, 37], information retrieval [7, 21, 23], and related areas [42]. After encoding high-dimensional feature vectors of documents, images, videos, or other types of data to compact binary codes, an effective binary coding or hashing method is expected to accomplish efficient similarity search while preserving the similarities among original data to some extent. As a result, using binary codes to represent and search in massive data is a promising solution to handle large-scale vision tasks, because of not only the storage efficiency (typically several hundred binary bits per data item) but also the high time efficiency of pairwise distance computations in a binary Hamming space. Besides hashing, vector quantization (VQ) [2, 6, 10, 27, 40, 41] is another promising direction for large-scale search problems in high-dimensional space. However, VQ and hashing are two very different techniques, especially in the search step. In this work, we focus on the latter one.

The binary coding or hashing techniques can be roughly divided into two major categories: *data-independent* and *data-dependent* methods. Locality-Sensitive Hashing (LSH) [7] is one of the most popular *data-independent* methods, which generates randomized hash functions via random projections. The LSH family has been continuously developed to accommodate a variety of distance and similarity measures, such as Euclidean distance, p-norm distance [3], Mahalanobis distance [15], kernel similarity [14, 28]. Although LSH is ensured to have high collision probability for similar data items, in practice LSH usually needs long hash bits and multiple hash tables to achieve both high precision and recall. The huge storage overhead may restrict its applications.

The other category, *data-dependent* methods (or *learning-based* methods in the literature), has witnessed a rapid development in the most recent years. The literature was comprehensive reviewed in [35] recently. Its emergence is due to the benefit that *learned* compact binary codes can effectively and efficiently index and organize massive data. Different from LSH, data-dependent binary coding methods aim to generate short binary codes using available training data. Among them, linear coding functions formed by a set of learned hyperplanes are mostly adopted, since they are computationally simple and easy to connect to traditional dimensionality reduction techniques. A number of algorithms in this category have been proposed, including unsupervised PCA Hashing [33], Iterative Quantization (ITQ) [8], Isotropic Hashing (IsoHash) [12], *etc*., and supervised Minimal Loss Hashing (MLH) [25, 26], Semi-Supervised Hashing (SSH) [33], Ranking-Based Su-

*Correspondence should be addressed to Fumin Shen (fumin.shen@gmail.com) or Wei Liu(wliu.cu@gmail.com).

pervised Hashing [34], FastHash [17], *etc*.

Compared to the above linear binary coding/hashing algorithms, nonlinear algorithms may generate more effective binary codes for data residing in nonlinear structures. Representative algorithms construct coding functions in a kernel space, for example, Binary Reconstructive Embedding (BRE) [13], Random Maximum Margin Hashing (RMMH) [11], Kernel-Based Supervised Hashing (KSH) [19], the kernel variant of ITQ [8], *etc*. It has also been shown that harnessing nonlinear manifold structures will help produce neighborhood-preserving compact binary codes. Spectral Hashing (SH) [37, 36] is a well-known algorithm in this style. More recently, Anchor Graph Hashing (AGH) [20, 18] leverages anchor graphs for making hash code training and out-of-sample extension to novel data both tractable and efficient. Shen *et al*. [30, 31] proposed a general Inductive Manifold Hashing (IMH) scheme which generates nonlinear coding functions by exploiting the flexibility of available manifold learning approaches.

Almost all the previous binary coding and hashing algorithms were designed to deal with Approximate Nearest Neighbor (ANN) search. Studies have rarely been dedicated to Maximum Inner Product Search (MIPS), which actually plays a critical role in various vision and learning applications [32]. The efficacy of aforementioned methods have not been validated on the MIPS problem yet. Shrivastava and Li [32] proposed an Asymmetric Locality-Sensitive Hashing (ALSH) algorithm for searching with (un-normalized) inner product being the underlying similarity measure. ALSH converts the MIPS problem to a standard LSH ANN problem by performing a simple asymmetric transformations on data pairs. While ALSH inherits all the theoretical guarantees of LSH [32], it can hardly achieve promising search performance using short binary codes.

In this work, we focus on learning data-dependent binary codes for tackling the MIPS problem. Inspired by the latest advance in asymmetric hashing[1], we propose an asymmetric binary code learning method for MIPS, thus named **Asymmetric Inner-product Binary Coding (AIBC)**. Specifically, two sets of coding functions are learned such that the inner products between their generated binary codes can reveal the inner products between original data vectors. Despite conceptually simple, the associated optimization is very challenging due to the highly nonsmooth nature of the objective that involves sign functions. To this end, we tackle the nonsmooth optimization in an alternating manner, by which a single coding function is solved with the others fixed. Through introducing auxiliary discrete variables to replace the sign functions, the optimization procedure is made efficient. As a simplified version of the proposed binary code learning framework, we propose another objective which maximizes the correlations between the inner products of the produced binary codes and raw data vectors. In both objectives, the binary codes and coding functions are simultaneously learned without continuous relaxations, which is the key to achieving high-quality binary codes.

The main contributions of our work are summarized as follows:

1. We propose a binary code learning framework for addressing the MIPS problem, which has the clear aim of preserving the inner-product similarities among raw data vectors. To this end, we design a tractable discrete optimization method, by which high-quality binary codes are iteratively generated with a closed-form solution for each bit.

2. To further speeding up binary code learning for MIPS, we propose an alternative simpler objective which maximizes the correlations between the inner products of the yielded binary codes and raw data vectors. By this objective, the binary codes can be learned in a much more efficient way and, usually, with higher quality. This is mainly because the problem can be easily solved with closed-form solutions for the two associated sub-problems.

3. Our two binary coding methods are extensively evaluated on several large-scale image datasets. The experimental results demonstrate the superiority of our methods over the state-of-the-arts.

The rest of this paper is organized as follows. Section 2 elaborates the details of the proposed AIBC methods. In Section 3, we evaluate our approaches on three real-world large-scale datasets, followed by the conclusion of this work in Section 4.

2. Asymmetric Inner-product Binary Coding

In this section, we present our binary code learning framework. We first give a brief introduction of MIPS; and then propose an inner-product fitting model to learn two asymmetric coding functions. To speed up the optimization, we practically simplify the original objective by maximizing inner-product correlation.

2.1. Maximum Inner Product Search (MIPS)

MIPS has been playing a significant role in various applications, such as recommender systems, deformable part model, multi-class classification [32]. Given a new query $\mathbf{q}$, MIPS targets at retrieving the datum having the largest inner product with $\mathbf{q}$ from the database $\mathbf{A}$. Formally, the

[1] An asymmetric LSH algorithm with sketches [5] was previously proposed at a early stage. Very recently, [24] discussed the power of asymmetric hashes.

MIPS problem is formulated as below:

$$\mathbf{p} = \arg\max_{\mathbf{a}\in\mathbf{A}} \ \mathbf{a}^\top \mathbf{q}. \tag{1}$$

For large-scale similarity search problems, it is practical to implement MIPS by employing binary coding techniques to achieve both storage and computational efficiencies. A binary coding function $h(\mathbf{x})$ maps an original feature vector $\mathbf{x}$ to a binary code of r bits in the Hamming space $\{1, -1\}^r$ [2]. Then problem (1) is reformulated as follows

$$\mathbf{p} = \arg\max_{\mathbf{a}\in\mathbf{A}} \ h(\mathbf{a})^\top h(\mathbf{q}). \tag{2}$$

It is natural to apply the popular locality sensitive hashing (LSH) for this problem. The LSH algorithm is simply constructed by the random projections generated from the standard normal distribution. Despite the popularity, the direct use of LSH on MIPS does not inherit the high collision probability guarantee of that on near neighbor search problems. To solve this problem, Shrivastava and Li [32] proposed the Asymmetric Locality Sensitive Hashing (ALSH) method, which converted the MIPS problem to the standard ℓ_2 nearest neighbor search problem. ALSH adopts two sets of *different* hash functions $h(\cdot)$ and $z(\cdot)$ to compute the inner product of the query and database point. The MIPS problem is performed as

$$\mathbf{p} = \arg\max_{\mathbf{a}\in\mathbf{A}} h(\mathbf{a})^\top z(\mathbf{q}), \tag{3}$$

where $h(\cdot)$ and $z(\cdot)$ are both constructed directly from LSH, by simply appending a few entries to $\mathbf{a}$ and $\mathbf{q}$ with different values. ALSH was proved to share the similar collision guarantee of similar points on MIPS as LSH on nearest neighbor search problem, which is mainly benefited from the flexibility of asymmetric hash functions.

2.2. Inner-product fitting

While the LSH and ALSH methods guarantee search accuracy to some extent, promising performance can hardly be achieved with compact binary codes due to the independence of data. To overcome the above limitation, we propose to *learn* hash functions rather than random projection for coping with the MIPS task.

Suppose that we have two sets of points: $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]$, where $\mathbf{a}_i \in \mathbb{R}^{d\times 1}$ and $\mathbf{x}_i \in \mathbb{R}^{d\times 1}$. Denote the similarity matrix of $\mathbf{A}$ and $\mathbf{X}$ as $\mathbf{S} \in \mathbb{R}^{n\times m}$, of which the element $\mathbf{S}_{ij}$ defines the similarity of $\mathbf{a}_i$ and $\mathbf{x}_j$. We aim to learn two sets of binary codes for $\mathbf{A}$ and $\mathbf{X}$ respectively, the inner product of which can well approximate $\mathbf{S}$. Inspired by [32], we adopt the asymmetric form of hash functions in our binary code learning formulation. We consider the following problem with hash functions $h(\cdot)$ and $z(\cdot)$,

$$\min_{h,z} ||h(\mathbf{A})^\top z(\mathbf{X}) - \mathbf{S}||^2. \tag{4}$$

Here $||\cdot||$ is the Frobenius norm. We will show next that the utilization of asymmetric hash functions can significantly facilitate the optimization of the above discrete optimization problem.

In our implementation, we compute the similarity matrix $\mathbf{S}$ by inner product, $\mathbf{S} = \mathbf{A}^\top\mathbf{X}$. With this setting, (4) turns out to be a problem of *inner product fitting* with binary codes. We therefore name the proposed binary code learning method employing asymmetric coding functions as **Asymmetric Inner-product Binary Coding (AIBC)**. In practice, we normalize $\mathbf{S}_{ij}$ to be 1 (or 0) when it is over (or smaller than) a threshold.

With the linear form of hash functions, *i.e.*, $h(\mathbf{x}) = \mathrm{sgn}(\mathbf{W}^\top\mathbf{x})$ and $z(\mathbf{x}) = \mathrm{sgn}(\mathbf{R}^\top\mathbf{x})$, where $\mathbf{W}, \mathbf{R} \in \mathbb{R}^{d\times r}$, we then have

$$\min_{\mathbf{W},\mathbf{R}} ||\mathrm{sgn}(\mathbf{W}^\top\mathbf{A})^\top \mathrm{sgn}(\mathbf{R}^\top\mathbf{X}) - \mathbf{S}||^2. \tag{5}$$

It is easy to see that the above problem is highly non-convex and difficult (usually NP hard) to solve due to the discrete sign functions. A feasible solution is to relax the discrete constraint by omitting the sign function. The continuous relaxation methodology is widely applied for hash code learning. However, this approximation method may accumulate considerable quantization errors, which makes the final binary codes less effective.

In order to obtain high-quality hash functions, we keep the sign function in our formulation. By taking the advantage of the flexibility of asymmetric hash functions, for this discrete optimization problem, we can naturally choose to solve $\mathbf{W}$ and $\mathbf{R}$ in an alternating fashion, that is, solve for one variable each time while keeping the other one fixed. We thus first consider the following sub-problem with variable $\mathbf{W}$ by fixing $z(\mathbf{X}) = \mathbf{Z}$ in (4),

$$\min_{\mathbf{W}} ||\mathrm{sgn}(\mathbf{W}^\top\mathbf{A})^\top\mathbf{Z} - \mathbf{S}||^2. \tag{6}$$

In practice, we initialize $\mathbf{R}$ by PCA projections. Similarly, by fixing $h(\mathbf{A}) = \mathbf{H}$, we obtain the following sub-problem with variable $\mathbf{R}$,

$$\min_{\mathbf{R}} ||\mathbf{H}^\top \mathrm{sgn}(\mathbf{R}^\top\mathbf{X}) - \mathbf{S}||^2, \tag{7}$$

Problem (6) or (7) is still with the sign function and cannot be solved trivially by an off-the-shelf solver. We now present a scalable and tractable method to (6), and (7) can be accordingly solved in the same way. To conquer the optimization involving discrete function, we introduce an auxiliary variable $\mathbf{B} \in \{-1, 1\}^{r\times n}$ to separate the optimizing

[2] Note that we use (1,-1) bits for mathematical derivations, and use (1,0) bits for implementations of all referred binary coding and hashing algorithms.

variable $\mathbf{W}$ and the sign function. We will show next the introduction of the auxiliary variable $\mathbf{B}$ (binary codes for $\mathbf{A}$) is the key to simplify the optimization. We rewrite problem (6) as

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{W}} ||\mathbf{B}^\top\mathbf{Z}-\mathbf{S}||^2+\lambda||\mathbf{B}-\mathbf{W}^\top\mathbf{A}||^2 \\ &\text{s.t.} \quad \mathbf{B}\in\{-1,1\}^{r\times n}. \end{aligned} \tag{8}$$

The objective of (8) has a clear explanation: the first term minimizes the inner products fitting error by the learned binary codes; while the second term ensures the hash function $h(\mathbf{x})$ can well predict the target binary codes $\mathbf{B}$ with minimum quantization loss. The parameter λ serve a trade-off between these two loss terms.

In problem (8), given $\mathbf{B}$, it is easy to compute $\mathbf{W}$

$$\mathbf{W}=(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{B}^\top. \tag{9}$$

We then have the following problem w.r.t. $\mathbf{B}$,

$$\begin{aligned} &\min_{\mathbf{B}} ||\mathbf{B}^\top\mathbf{Z}||^2-2\text{trace}(\mathbf{B}^\top\mathbf{D}) \\ &\text{s.t.} \quad \mathbf{B}\in\{-1,1\}^{r\times n}, \end{aligned} \tag{10}$$

where $\mathbf{D}=\mathbf{Z}\mathbf{S}^\top+\lambda\mathbf{W}^\top\mathbf{A}$. Note that $||\mathbf{B}||^2=nr$.

For this key sub-problem, inspired by the recent progress of binary codes optimization [29], we choose to solve one row of $\mathbf{B}$ each time while fixing all other rows, *i.e.*, we compute one-bit for all n samples each time. Let $\mathbf{b}$ be the l^{th} row of $\mathbf{B}$, $l=1,\cdots,r$, and $\tilde{\mathbf{B}}$ the remaining rows of $\mathbf{B}$. Then $\mathbf{b}$ contains one bit for each of n samples. Similarly, let $\mathbf{d}$ be the l^{th} row of $\mathbf{D}$, $\tilde{\mathbf{D}}$ the matrix of $\mathbf{D}$ excluding $\mathbf{d}$, $\mathbf{z}$ the l^{th} row of $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$ the matrix of $\mathbf{Z}$ excluding $\mathbf{z}$. With these notations and a few simple matrix manipulations, problem (10) can be written as w.r.t. $\mathbf{b}$

$$\begin{aligned} &\min_{\mathbf{b}} \mathbf{b}(\tilde{\mathbf{B}}^\top\tilde{\mathbf{Z}}\mathbf{z}^\top-\mathbf{d}^\top) \\ &\text{s.t.} \quad \mathbf{b}\in\{-1,1\}^n. \end{aligned} \tag{11}$$

Thus, we obtain the optimal solution for the l^{th} row of $\mathbf{B}$,

$$\mathbf{b}=\text{sgn}(\mathbf{z}\tilde{\mathbf{Z}}^\top\tilde{\mathbf{B}}-\mathbf{d}). \tag{12}$$

By this method, each bit is iteratively updated with the prelearned $r-1$ bits till the procedure converges with a set of better codes. In our experiments, the procedure usually converges with less than 10 iterations. With the analytical solution for each bit, the whole optimization is very efficient and thus can easily scale to massive data. Till now, we are ready to solve problem (8) (therefor also (6)) iteratively with the solution of $\mathbf{W}$ and $\mathbf{B}$ provided above. By iteratively solve (6) and (7), we finally obtain a pair of hash function of $h(\cdot)$ and $z(\cdot)$. Although this method can hardly achieve the globally optimal solution for the discrete optimization problem, at each step, the local optimal solution for each variable (*e.g.*, $\mathbf{W}$, $\mathbf{b}$) is obtained in a closed form. We will show in the experiments that this optimization strategy works very well. Since the objective (4) of this method involves a quadratic term with the hash functions, we denote this method as AIBC-Q.

Note that the supervised hashing approach KSH [19] used a similar objective as in (4). However, our binary coding approach mainly differs from KSH in the following three aspects: 1) KSH is supervised while our AIBC is unsupervised. 2) For each binary bit, KSH learns a single coding function while AIBC learns two coding functions in an asymmetric fashion. 3) KSH applies a greedy optimization procedure to solve a relaxed problem by using a sigmoid function to replace the sign function. In contrast, AIBC directly optimizes the binary codes without resorting to any continuous relaxations. In the experiments, we implement an unsupervised version of KSH and the comparative results clearly show the advantage of our AIBC technique.

2.3. Inner-product correlation maximization

In this part, we propose an alternative model to further speed up the learning procedure of AIBC. The new model simplifies the objective in (4) by directly maximizing the correlation of the similarity matrix $\mathbf{S}$ and $\big(h(\mathbf{A})^\top z(\mathbf{X})\big)$,

$$\max_{h,z} \text{trace}(h(\mathbf{A})\mathbf{S}z(\mathbf{X})^\top). \tag{13}$$

We note that the objective function of (13) can be easily obtained by discarding the quadratic term $||h(\mathbf{A})^\top z(\mathbf{X})||^2$ in (4). Note that the quadratic term does not leverage the groundtruth similarity and can be seen as a regularization with the magnitude of the learned inner product. We find in practice omitting this term does not adversely affect the performance. In contrast, we show next it provides a much more efficient mean of solving the asymmetric hash function learning problem.

Using the same strategy described in the previous section, for (13) we obtain the following two sub-problems (14) and (15) w.r.t. $\mathbf{W}$ and $\mathbf{R}$, respectively.

$$\max_{\mathbf{W}} \text{trace}(\text{sgn}(\mathbf{W}^\top\mathbf{A})\mathbf{S}\mathbf{Z}^\top); \tag{14}$$

$$\max_{\mathbf{R}} \text{trace}(\mathbf{H}\mathbf{S}\text{sgn}(\mathbf{R}^\top\mathbf{X})^\top). \tag{15}$$

By introducing the auxiliary variable $\mathbf{B}$, problem (14) is formulated as

$$\begin{aligned} &\max_{\mathbf{B},\mathbf{W}} \text{trace}(\mathbf{B}\mathbf{S}\mathbf{Z}^\top)-\lambda||\mathbf{B}-\mathbf{W}^\top\mathbf{A}||^2 \\ &\text{s.t.} \quad \mathbf{B}\in\{-1,1\}^{r\times n}. \end{aligned} \tag{16}$$

Same as problem (8), we solve problem (16) iteratively with $\mathbf{B}$ and $\mathbf{W}$. The difference is, problem (16) benefits

Table 1: Results in terms of mAP, Precision of top 500 samples and Hamming distance 2 Precision of the compared methods on **SUN397** with 32, 64 and 128 bits, respectively.

Method	mAP			Precision@500			HD2 Precision		
	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit
LSH	0.0605	0.0920	0.1428	0.0921	0.1397	0.2036	0.0304	0.0011	0.0002
ALSH	0.0592	0.0913	0.1425	0.0918	0.1406	0.2033	0.0352	<0.0001	<0.0001
SH	0.2103	0.2191	0.1991	0.2737	0.2877	0.2735	0.2479	0.0084	0.0002
IsoHash	0.2414	0.2668	0.2850	0.2912	0.3198	0.3391	0.3101	0.0131	0.0005
ITQ	0.3014	0.3336	0.3456	0.3448	0.3774	0.3887	0.3815	0.0779	0.0039
InnerKSH	0.2900	0.3312	0.3558	0.3367	0.3734	0.3923	0.3735	0.1442	0.0314
AIBC-Q	0.3549	0.3925	0.4299	0.3996	0.4263	0.4409	0.4684	0.1646	0.0376
AIBC-L	**0.3639**	**0.4042**	**0.4348**	**0.4078**	**0.4428**	**0.4642**	**0.4704**	**0.1746**	**0.0522**

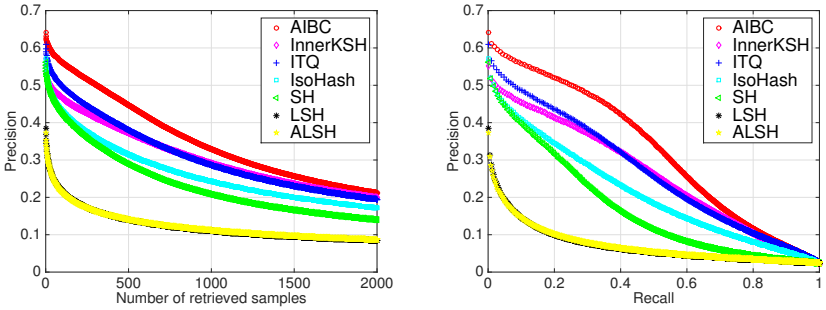

Figure 1: Precision curve of up to top 2000 retrieved samples and precision-recall curves on SUN397. We only report AIBC-L for AIBC for clarity. 64 bits are used.

from the advantage that it has a optimal analytical solution for $\mathbf{B}$ with a given $\mathbf{W}$,

$$\mathbf{B} = \operatorname{sgn}\left(\mathbf{Z}\mathbf{S}^\top + 2\lambda\mathbf{W}^\top\mathbf{A}\right). \tag{17}$$

This property does not only make (16) much more efficient to solve but also provide more accurate solution for the whole optimization. This explains why this method performs slightly better than AIBC-Q in our experiments. The optimization of (16) can be easily solved by iteratively updating $\mathbf{B}$ and $\mathbf{W}$ (by (9)). With the above closed-form solutions, the training of the proposed method can be easily performed on large-scale data with very high efficiency. Since the objective (13) of this method only involves a linear term with each hash function, we denote this method as AIBC-L.

The proposed Asymmetric Inner-product Binary Coding (AIBC) is summarized in Algorithm 1.

Algorithm 1 Asymmetric Inner-product Binary Coding (AIBC)

Input: Data $\mathbf{A}$ and $\mathbf{X}$; code length r; maximum iteration number t; parameters λ.
Output: Hash function $h(\mathbf{x})$ and $z(\mathbf{x})$.

1. Compute the similarity matrix $\mathbf{S} = \mathbf{A}^\top\mathbf{X}$.
2. Initialize $\mathbf{R}$ by PCA projections.
3. Loop until converge or reach maximum t iterations:
   - h-step: Compute $\mathbf{W}$ by solving problem (6) or (14).
   - z-step: Compute $\mathbf{R}$ by solving problem (7) or (15).

3. Experiments

In this section, we evaluate the proposed methods on three large-scale datasets: YouTube Faces [38], SUN397 [39] and ImageNet [4]. The proposed two methods AIBC-Q and AIBC-L are compared against several state-of-the-art hashing methods including LSH (implemented by signed random projections), ALSH [32], SH [37], ITQ [8], IsoHash [12]. We also compare the unsupervised version of KSH [19] implemented by ourselves with the similarity computed by inner product (thus named InnerKSH). We use the public codes and suggested parameters of these methods from the authors. For InnerKSH, we use 4,000 training samples to form the pairwise similarity matrix and randomly choose 1,000 samples as anchor points. Since our methods are unsupervised, we do not compare them with the supervised methods. For our AIBC, we empirically set the parameter λ to 100 and the maximum iteration number $t = 2$; the data matrix $\mathbf{A}$ is set as the whole training data; $\mathbf{X}$ is formed by the 10,000 randomly selected points from training data. We binarize each column of $\mathbf{S}$ with a threshold

Table 2: Results in terms of mAP, Precision of top 500 samples and Hamming distance 2 Precision of the compared methods on the **YouTube Faces** database with 32, 64 and 128 bits, respectively.

Method	mAP			Precision@500			HD2 Precision		
	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit
LSH	0.1341	0.2513	0.4092	0.2710	0.4811	0.7004	0.4444	0.2388	0.1400
ALSH	0.1124	0.2104	0.3565	0.2210	0.4106	0.6355	0.3262	0.0726	< 0.0001
SH	0.6543	0.6395	0.5677	0.8556	0.9001	0.9047	0.9596	0.3547	0.1447
IsoHash	0.6756	0.7204	0.7274	0.8698	0.9150	0.9274	0.9561	0.5037	0.1492
ITQ	0.7443	0.7775	0.7767	0.8979	0.9321	0.9416	0.9646	0.8229	0.4587
InnerKSH	0.7512	0.7634	0.7757	0.8989	0.9138	0.9239	0.9697	0.8676	0.6434
AIBC-Q	0.7913	0.8175	0.8293	0.9393	0.9502	**0.9625**	0.9859	0.9444	0.8532
AIBC-L	**0.8152**	**0.8233**	**0.8400**	**0.9477**	**0.9551**	0.9590	**0.9931**	**0.9716**	**0.9329**

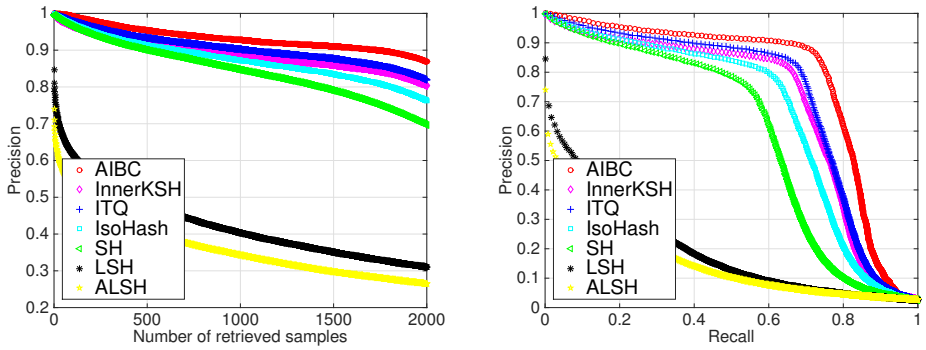

Figure 2: Precision curve of up to top 2000 retrieved samples and precision-recall curves on **YouTube Faces**. We only report AIBC-L for AIBC for clarity. 64 bits are used.

computed by the k_{th} largest one among the n inner products, where k is set to 500 for SUN397, 1000 for ImageNet and 2000 for YouTube Faces. For evaluation, database data and queries are compressed by $h(\mathbf{x})$ and $z(\mathbf{x})$, respectively.

We report the compared results in terms of both hash lookup: precision of Hamming distance 2 (HD2 Precision) and Hamming ranking: mean of average precision (mAP) and mean precision of the top 500 retrieved neighbors (Precision@500). We also present the detailed results by precision-recall and the precision of top 2000 curves. Note that we treat a query a false case if no point is returned when calculating precisions. Ground truths are defined by the category information from the datasets.

3.1. SUN397: retrieval with scene images

SUN397 [39] contains about 108K images from 397 scene categories, where each image is represented by a 1,600-dimensional feature vector extracted by PCA from 12,288-dimensional Deep Convolutional Activation Features [9]. We use a subset of this dataset including 42 categories with each containing more than 500 images (with total 33K images); 100 images are sampled uniformly randomly from each category to form a test set of 4,200 images.

The comparative results are shown in Table 1. First we can see that the proposed AIBC-L significantly outperforms all other algorithms in mAP, precision of top 500 and Hamming distance 2 precision. Our AIBC-Q performs slight worse than AIBC-L on this dataset, although it still outperforms all other methods in all situations. Among other compared methods, ITQ and InnerKSH achieve best results. It is not surprising that the data-independent algorithm LSH and ALSH do not perform as well as other learning based methods. Interestingly we also observe that the asymmetric ALSH algorithm achieves close results with the original signed random projection based LSH method on this dataset. The detailed precision curves of top 2000 retrieved samples and the precision-recall curves using 64 bits are shown in Figure 1. We can easily see that the performance rank of the precision curves of the compared methods is consistent with the above analysis. The proposed AIBC still performs the best among the compared algorithms.

Efficiency We further take the SUN397 dataset as an example to evaluate the computational efficiency of these compared algorithms. The model training time and the testing time (of compressing one query into binary codes with the trained model) are shown in Table 4. First, we are not surprised to see that the proposed AIBC-L has a significant

Table 3: Results in terms of mAP, Precision of top 500 samples and Hamming distance 2 Precision of the compared methods on the **ImageNet** database with 32, 64 and 128 bits, respectively.

Method	mAP			Precision@500			HD2 Precision		
	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit
LSH	0.0496	0.0974	0.1743	0.1036	0.1963	0.3133	0.0708	< 0.0001	< 0.0001
ALSH	0.0495	0.0907	0.1694	0.1043	0.1847	0.3074	0.0644	< 0.0001	< 0.0001
SH	0.2418	0.3066	0.3309	0.3647	0.4531	0.4956	0.4029	0.0327	0.0004
IsoHash	0.2521	0.3326	0.3847	0.3673	0.4649	0.5231	0.4194	0.0342	0.0002
ITQ	0.3231	0.4127	0.4621	0.4304	0.5313	0.5882	0.4619	0.1730	0.0435
InnerKSH	0.4073	0.4651	0.4850	0.5080	0.5693	0.5893	0.4942	0.2345	0.0743
AIBC-Q	0.4421	0.5280	**0.5756**	0.5702	0.6180	**0.6658**	0.5280	0.3102	0.1828
AIBC-L	**0.4771**	**0.5402**	0.5753	**0.5796**	**0.6375**	0.6643	**0.5595**	**0.3817**	**0.2243**

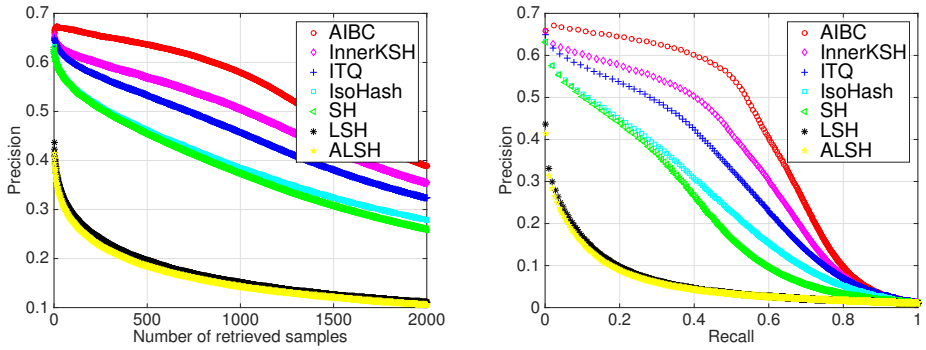

Figure 3: Precision curve of up to top 2000 retrieved samples and precision-recall curves on **ImageNet**. We only report AIBC-L for AIBC for clarity. 64 bits are used.

computational advantage over AIBC-Q, due to the simplified objective. Compared to SH, IsoHash and ITQ, AIBC-L consumes more time to train the hash functions, which is mainly occupied by the inner product matrix calculation. However, the training of AIBC-L is still sufficiently efficient to scale to large-scale data: it runs only about 15 seconds on a standard PC for training with the whole 33K images of the SUN397 database. As can be seen, InnerKSH suffers from a huge computational overhead with the greedy optimization procedure, while AIBC-L can be trained much more efficiently with the closed-form solution of each subproblem.

3.2. YouTube Faces: retrieval with face images

YouTube Faces dataset contains 1,595 different people, from which we choose 340 people such that each one has at least 500 images to form a subset of 370,319 face images, and represent each face image as a 1,770-dimensional LBP feature vector [1]. We use a subset of YouTube Faces with 38 people each containing more than 2,000 faces (about 100K images in total). The test set includes 3,800 face images which are evenly sampled from each of the 38 classes.

We report the results on YouTube Faces in Table 2. Again, the proposed AIBC-L and AIBC-Q achieve the best results on this dataset in all situations. For instance, with 64 bits the proposed AIBC-L obtains 82.33% which are higher than the second best 77.75% (ITQ) by 4.85%. SH and IsoHash do not perform as well as other data-dependent methods on this dataset. For HD2 Precision, we can see that most of the data-dependent methods perform well with short code lengths. However, SH, IsoHash and ITQ suffer from dramatic performance drop with long hash bits due to the increasingly sparsity of Hamming space. In contrast, with 128

Table 4: Comparison of the the computational efficiency (in consumed seconds) on SUN397. Both the training and testing time are compared with two different code lengths.

Method	Training time		Testing time	
	64-bit	128-bit	64-bit	128-bit
LSH	0.0846	0.1459	3.136e-6	3.629e-6
ALSH	0.5460	0.5976	1.296e-5	1.486e-5
SH	2.377	5.223	2.750e-5	1.060e-4
IsoHash	1.825	2.311	3.664e-6	6.335e-6
ITQ	3.326	5.296	3.527e-6	6.409e-6
InnerKSH	1933.1	4259.1	5.533e-5	6.1706e-5
AIBC-Q	53.29	173.9	3.179e-6	4.240e-6
AIBC-L	15.19	15.61	3.272e-6	3.853e-6

bits AIBC-L still achieves a high precision (93.29%) which outperforms all other methods by large margins. The superiority of the proposed AIBC is further shown in Figure 2. It is clear that AIBC ranks the first on both precision of top 2000 and precision-recall.

3.3. ImageNet: retrieval with large dataset

As a subset of ImageNet [4], the large dataset ILSVRC 2012 contains over 1.2 million images of totally 1,000 categories. We form the retrieval database by the 100 largest classes with total 128K images from the provided training set, and 50,000 images from the validation set as the query set. As in [17], we use the 4096-dimensional features extracted by the convolution neural networks (CNN) model.

The results are shown in Table 3. The superiority of the AIBC is further demonstrated on this large-scale database. For example, with 64-bit AIBC-L achieves 54.02% mAP while the best results of other methods is 46.51% obtained by InnerKSH. For hash lookup, AIBC-L gets a precision of 22.43% while those of ITQ and InnerKSH are all less than 10%. Between the proposed methods, AIBC-Q tends to achieve better MAP and precision than AIBC-L with long binary codes. However, in HD2 precision AIBC-Q is still inferior to AIBC-L. Figure 3 illustrates the precision and precision-recall curves, which clearly show AIBC performs better than other methods on this large dataset.

3.4. Algorithm analysis

In this subsection, we empirically evaluate the proposed algorithm (AIBC-L) on ImageNet with parameter λ varied from 0.1 to 10,000. We observe from Figure 4 (left) that the performance reaches the peak at $\lambda = 100$, and drops dramatically with larger λs. We also show in Figure 4 (right) the objective values of our algorithm with increasing number of iterations. As can be seen, the objective value does not change significantly with more than 5 iterations.

4. Conclusions

This paper focused on binary code learning for the MIPS problem through proposing an inner-product fitting framework. In the framework, two asymmetric coding functions are learned such that the inner products between original data pairs are approximated by the produced binary code vectors. While this framework is conceptually simple, the associated optimization is very challenging due to the presence of discrete sign functions. Benefiting from the flexibility of the asymmetric coding mechanism, we solved the optimization in an alternating fashion involving two subproblems. We also proposed an alternative objective that maximizes the correlations between the inner products of the pursed binary codes and raw data vectors, which enjoys a closed-form solution for each sub-problem, thus making the whole optimization more efficient.

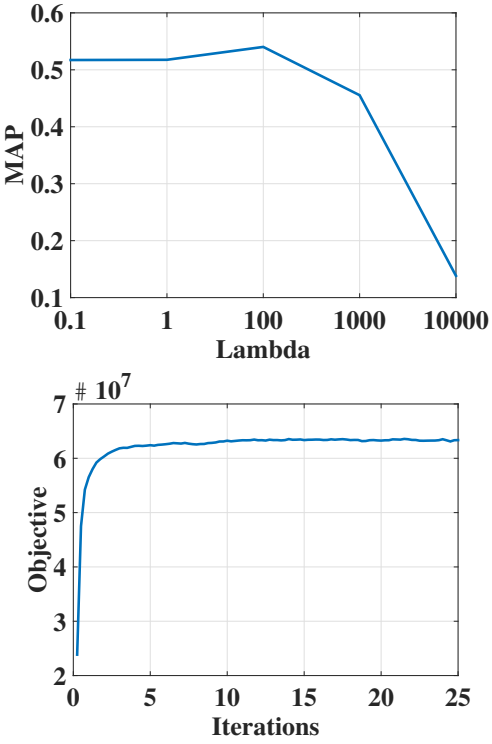

Figure 4: (Top) MAPs with varying λs; (Bottom) the function of objective value with iterations.

To obtain high-quality binary codes, both of our proposed two approaches directly optimize the target binary codes without resorting to continuous relaxations. The proposed Asymmetric Inner-product Binary Coding (AIBC) technique with both qudratic (AIBC-Q) and linear objectives (AIBC-L) were evaluated on several large-scale image datasets. Experimental results showed AIBC significantly outperforms several state-of-the-art hashing methods.

References

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE TPAMI*, 28(12):2037–2041, 2006. 7

[2] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In *Proc. CVPR*, pages 931–938, 2014. 1

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

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 5, 8

[5] W. Dong, M. Charikar, and K. Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *Proc. SIGIR*, pages 123–130, 2008. 2

[6] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2946–2953, 2013. 1

[7] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. VLDB*, 1999. 1

[8] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE TPAMI*, 35(12):2916–2929, 2013. 1, 2, 5

[9] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proc. ECCV*, pages 392–407. Springer, 2014. 6

[10] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 33(1):117–128, 2011. 1

[11] A. Joly and O. Buisson. Random maximum margin hashing. In *Proc. CVPR*, 2011. 2

[12] W. Kong and W.-J. Li. Isotropic hashing. In *Proc. NIPS*, 2012. 1, 5

[13] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Proc. NIPS*, 2009. 2

[14] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Proc. ICCV*, 2009. 1

[15] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE TPAMI*, 31(12):2143–2157, 2009. 1

[16] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. Dick. Learning hash functions using column generation. In *Proc. ICML*, 2013. 1

[17] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proc. CVPR*, 2014. 2, 8

[18] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS 27*, 2014. 2

[19] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proc. CVPR*, 2012. 1, 2, 4, 5

[20] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proc. ICML*, 2011. 2

[21] X. Liu, J. He, C. Deng, and B. Lang. Collaborative hashing. In *Proc. CVPR*, pages 2147–2154, 2014. 1

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

[23] X. Liu, L. Huang, C. Deng, J. Lu, and B. Lang. Multi-view complementary hash tables for nearest neighbor search. In *Proc. ICCV*, 2015. 1

[24] B. Neyshabur and N. Srebro. On symmetric and asymmetric lshs for inner product search. In *Proc. ICML*, pages 1926–1934, 2015. 2

[25] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *Proc. ICML*, 2011. 1

[26] M. Norouzi, D. M. Blei, and R. R. Salakhutdinov. Hamming distance metric learning. In *Proc. NIPS*, 2012. 1

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

[28] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Proc. NIPS*, 2009. 1

[29] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *Proc. CVPR*, pages 37–45, 2015. 4

[30] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang. Inductive hashing on manifolds. In *Proc. CVPR*, 2013. 2

[31] F. Shen, C. Shen, Q. Shi, A. van den Hengel, Z. Tang, and H. T. Shen. Hashing on nonlinear manifolds. *IEEE TIP*, 24(6):1839–1851, 2015. 2

[32] A. Shrivastava and P. Li. Asymmetric lsh (ALSH) for sublinear time maximum inner product search (MIPS). In *Proc. NIPS*, pages 2321–2329. 2014. 2, 3, 5

[33] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large scale search. *IEEE TPAMI*, 34(12):2393–2406, 2012. 1

[34] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang. Learning hash codes with listwise supervision. In *Proc. ICCV*, 2013. 2

[35] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014. 1

[36] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *Proc. ECCV*, 2012. 2

[37] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. NIPS*, 2008. 1, 2, 5

[38] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proc. CVPR*, pages 529–534, 2011. 5

[39] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, pages 3485–3492, 2010. 5, 6

[40] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *Proc. ICML*, pages 838–846, 2014. 1

[41] T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Sparse composite quantization. In *Proc. CVPR*, pages 4548–4556, 2015. 1

[42] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen. Sparse hashing for fast multimedia search. *ACM TOIS*, 31(2):9:1–9:24, 2013. 1