Supervised Discrete Hashing

Fumin Shen* Chunhua Shen† Wei Liu⋆ Heng Tao Shen‡
* University of Electronic Science and Technology of China ⋆ IBM Research
† University of Adelaide; and Australian Centre for Robotic Vision ‡ The University of Queensland

Abstract

Recently, learning based hashing techniques have attracted broad research interests because they can support efficient storage and retrieval for high-dimensional data such as images, videos, documents, etc. *However, a major difficulty of learning to hash lies in handling the discrete constraints imposed on the pursued hash codes, which typically makes hash optimizations very challenging (NP-hard in general). In this work, we propose a new supervised hashing framework, where the learning objective is to generate the optimal binary hash codes for linear classification. By introducing an auxiliary variable, we reformulate the objective such that it can be solved substantially efficiently by employing a regularization algorithm. One of the key steps in this algorithm is to solve a regularization sub-problem associated with the NP-hard binary optimization. We show that the sub-problem admits an analytical solution via cyclic coordinate descent. As such, a high-quality discrete solution can eventually be obtained in an efficient computing manner, therefore enabling to tackle massive datasets. We evaluate the proposed approach, dubbed Supervised Discrete Hashing (SDH), on four large image datasets and demonstrate its superiority to the state-of-the-art hashing methods in large-scale image retrieval.*

1. Introduction

Hashing has attracted considerable attention of researchers in computer vision, machine learning, information retrieval and related areas [8,10,16,20,21,31,34,38,40]. Hashing techniques encode documents, images, videos or other sorts of data by a set of short binary codes, while preserving the similarity of the original data. With the binary codes, the task of nearest neighbour search can be easily conducted on large-scale dataset, due to the high efficiency of pairwise comparison with the Hamming distance.

Among various hashing techniques, Locality-Sensitive Hashing (LSH) [8] is one of the most popular *data-independent* methods, which generates hash functions by random projections. In addition to traditional Euclidean distance, LSH has been generalized to accommodate other distance and similarity measures such as p-norm distance [4], Mahalanobis metric [15], and kernel similarity [14, 27]. A disadvantage of the LSH family is that LSH usually needs long bit length (≥ 1000) to achieve both high precision and recall. This leads to a huge storage overhead and thus limits the sale at which an LSH algorithm may be applied.

Recently, learning-based *data-dependent* hashing methods have become increasingly popular because of the benefit that *learned* compact binary codes can effectively and highly efficiently index and organize massive data. Instead of constructing hash functions randomly like LSH, data-dependent hashing methods aim to generate short hash codes (typically ≤ 200) using available training data. Various hashing algorithms have been proposed in the literature, of which a large category focuses on linear hashing algorithms which learn a set of hyperplanes as linear hash functions. The representative algorithms in this category include unsupervised PCA Hashing [34], Iterative Quantization (ITQ) [10], Isotropic Hashing [12], *etc*., and supervised Minimal Loss Hashing (MLH) [24, 25], Semi-Supervised Hashing (SSH) [34], LDA Hashing [2], Ranking-Based Supervised Hashing [35], FastHash [17], *etc*. A bilinear form of hash functions were introduced by [9, 22].

As an extension of linear hash functions, a variety of algorithms have been proposed to generate nonlinear hash functions in a kernel space, including Binary Reconstructive Embedding (BRE) [13], Random Maximum Margin Hashing (RMMH) [11], Kernel-Based Supervised Hashing (KSH) [20], the kernel variant of ITQ [10], *etc*. In parallel, harnessing nonlinear manifold structures has been shown effective in producing compact neighborhood-preserving hash codes. The early algorithm in this fashion is Spectral Hashing (SH) [37, 38], which produces hash codes through solving a continuously relaxed mathematical program similar to Laplacian Eigenmaps [1]. More recently, Anchor Graph Hashing (AGH) [19, 21] leveraged anchor graphs for solving the eigenfunctions of the resulting graph Laplacians, making hash code training and out-of-sample extension to novel data both tractable and efficient for large-scale datasets. Shen *et al*. [29, 30] proposed a general Inductive

Manifold Hashing (IMH) scheme that also generates nonlinear hash functions.

In general, the discrete constraints imposed on the binary codes that the target hash functions generate lead to mixed-integer optimization problems - which are generally NP-hard. To simplify the optimization involved in a binary code learning procedure, most of the aforementioned methods chose to first solve a relaxed problem through discarding the discrete constraints, and then threshold (or quantize) the solved continuous solution to achieve the approximate binary solution. This relaxation scheme greatly simplifies the original discrete optimization. Unfortunately, such an approximate solution is typically of low quality and often makes the resulting hash functions less effective possibly due to the accumulated quantization error, which is especially the case when learning long-length codes.

Directly learning the binary codes without relaxations would be preferable if (and only if) a tractable and scalable solver is available. The importance of discrete optimization in hashing has been rarely taken into account by most existing hashing methods. Iterative Quantization (ITQ) [10] is an effective approach to decrease the quantization error by applying an orthogonal rotation to projected training data. One limitation of ITQ is that it learns orthogonal rotations over pre-computed mappings (*e.g*., PCA or CCA). The separate learning procedure usually makes ITQ suboptimal.

In this work, we propose a novel supervised hashing framework, which aims to directly optimize the binary hash codes *effectively* and *efficiently*. To leverage supervised label information, we formulate the hashing framework in terms of linear classification, where the learned binary codes are expected to be optimal for classification. More specifically, the learned binary codes can be viewed as nonlinearly generated feature vectors of original data. The label information is exploited so that these binary feature vectors are easy to be classified. Similar to discrete boosting learning at the high level, we nonlinearly transform the original data into a binary space, and then classify the original data in this space.

To fulfill this idea, we propose a joint optimization procedure which jointly learns a binary embedding and a linear classifier. In this formulation, a group of hash functions are simultaneously optimized to fit the learned binary bits. To better capture the nonlinear structure underlying input data, those hash functions are learned in a kernel space. The entire joint optimization is then executed in an iterative manner with three associated sub-problems.

To solve the most critical sub-problem - binary code optimization, in our supervised hashing framework we propose a *discrete cyclic coordinate descent* (DCC) algorithm to generate the hash codes bit by bit. By carefully choosing loss functions for the linear classifier, the DCC algorithm yields the *optimal* hash bits in a closed form, which consequently makes the entire optimization procedure very efficient and naturally scale to massive datasets. We name the proposed supervised hashing approach employing discrete cyclic coordinate descent as *Supervised Discrete Hashing* (SDH). Our main contributions are summarized as follows:

1. We propose a novel supervised hashing approach based on the assumption that good hash codes are optimal for linear classification. Our key technique lies in directly solving the corresponding discrete optimization without any relaxations. First, by introducing an auxiliary variable, we reformulate the optimization objective such that it can be solved efficiently using a regularization scheme. Second, a key step of our SDH approach is to solve the NP-hard binary optimization sub-problem. By means of discrete cyclic coordinate descent, at each step SDH solves the associated binary optimization and obtains an analytical solution, which thus makes the whole optimization very efficient. We show that direct optimization of the discrete bits without relaxation plays critical roles in achieving high-quality hash codes.
2. The proposed SDH is evaluated on four large-scale benchmark datasets, and its efficacy is validated by the superior experimental results over several state-of-the-art hashing methods.

Most recently, the graph cuts algorithm was applied by FastHash [17] and GCC [7] to solve binary hash codes. In these two methods, each bit of the learned binary codes is used as classification labels to train a classifier. The similar idea was also adopted in [17, 18, 39]. One significant difference between our SDH and the graph-cut based methods is that we treat all the hash bits (constituting a hash code vector) generated for each sample as an input feature vector for the linear classifier. Concurrent to our work, Liu *et al*. [19] also realized the importance of the quality of binary optimization in hashing but focused on a completely different problem, *i.e*., unsupervised discrete hashing.

The code for the proposed SDH has been released at `https://github.com/bd622/DiscretHashing`.

2. Supervised Discrete Hashing

Suppose that we have n samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$. We aim to learn a set of binary codes $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^n \in \{-1, 1\}^{L \times n}$ to well preserve their semantic similarities, where the i^{th} column $\mathbf{b}_i$ is the L-bits binary codes for $\mathbf{x}_i$.

To take advantage of the label information, here we consider the binary codes learning problem in the framework of linear classification. That is, we expect the learned binary codes to be optimal for the jointly learned linear classifier. In other words, our hypothesis is that good binary codes are ideal for classification too.

We adopt the following multi-class classification formulation

$$\mathbf{y} = G(\mathbf{b}) = \mathbf{W}^\top \mathbf{b} = [\mathbf{w}_1^\top \mathbf{b}, \cdots, \mathbf{w}_C^\top \mathbf{b}]^\top \tag{1}$$

where $\mathbf{w}_k \in \mathbb{R}^{L\times 1}$, $k = 1, \cdots, C$ is the classification vector for class k and $\mathbf{y} \in \mathbb{R}^{C\times 1}$ is the label vector, of which the maximum item indicates the assigned class of $\mathbf{x}$.

We choose to optimize the following problem:

$$\begin{aligned} \min_{\mathbf{B},\mathbf{W},F} \quad & \sum_{i=1}^{n} L(\mathbf{y}_i, \mathbf{W}^\top \mathbf{b}_i) + \lambda ||\mathbf{W}||^2 \\ \text{s.t.} \quad & \mathbf{b}_i = \text{sgn}(F(\mathbf{x}_i)), i = 1, \cdots, n. \end{aligned} \tag{2}$$

Here $L(\cdot)$ is the loss function and λ is the regularization parameter; $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n \in \mathbb{R}^{C\times n}$ is the ground truth label matrix, where $y_{ki} = 1$ if $\mathbf{x}_i$ belongs to class k and 0 otherwise. The hash function $H(\mathbf{x}) = \text{sgn}(F(\mathbf{x}))$ encodes $\mathbf{x}$ by L bits. Here $\text{sgn}(\cdot)$ is the sign function, which outputs $+1$ for positive numbers and -1 otherwise. $||\cdot||$ is the ℓ_2 norm for vectors and Frobenius norm for matrices. We will discuss the form of the hash functions in the next section.

Here we want to emphasize that it is essential to have the discrete variable $\mathbf{b}_i$ ($i = 1, \cdots$). At the first glance, $\mathbf{b}_i$ is not of interest because we are interested in learning the hash functions $F(\cdot)$. In problem (2), one can simply remove the constraints by eliminating the auxiliary variables $\mathbf{b}_i$. Doing so leads to an optimization problem similar to BRE [13], which is very difficult and slow to optimize. We propose a significantly more efficient and effective optimization method by re-parameterizing the problem.

The problem (2) is in general still NP hard and difficult to solve with the discrete variable $\mathbf{b}_i$. One can always obtain an approximate solution by simply relaxing the binary constraint to be continuous $\mathbf{b}_i = F(\mathbf{x}_i)$. With this relaxation, the continuous embeddings $\mathbf{b}_i$ are first learned, which are then thresholded to be binary codes. Most existing hashing algorithms adopt this relaxation approach. Examples include Spectral Hashing [38], PCAH [34], AGH [21], IMH [29], *etc*. This approach usually make the original problem much easier to solve. However, clearly it is only sub-optimal as mentioned before.

In order to achieve binary codes of better quality, here we keep the binary constraints of $\mathbf{b}_i$ in the optimization problem and attempt to solve it much more efficiently. Inspired by the regularization methods in large-scale optimization [23][1], we rewrite problem (2) as

$$\begin{aligned} \min_{\mathbf{B},\mathbf{W},F} & \sum_{i=1}^{n} L(\mathbf{y}_i, \mathbf{W}^\top \mathbf{b}_i) + \lambda ||\mathbf{W}||^2 + \nu \sum_{i=1}^{n} ||\mathbf{b}_i - F(\mathbf{x}_i)||^2 \\ \text{s.t. } & \mathbf{b}_i \in \{-1, 1\}^L. \end{aligned} \tag{3}$$

The last term in (3) models the fitting error of the binary codes $\mathbf{b}_i$ by the continuous embedding $F(\mathbf{x}_i)$ and ν is the penalty parameter. In theory, with a sufficiently large ν, problem (3) becomes arbitrarily close to (2). In practice, small differences between $\mathbf{b}_i$ and $F(\mathbf{x}_i)$ are acceptable in our applications. The recent work [36] applied a similar formulation as (3). However, the former one did not impose the binary constraints, which thus made the solution distinct from the one presented in this paper.

It is easy to see that, the above joint optimization problem is still highly non-convex and difficult to solve. We will show that, however, it is tractable to solve the problem with respect to one variable while keeping other two variables fixed, given a proper loss function $L(\cdot)$. Naturally we can iteratively solve each variable in problem (3) one by one. Next let us first define the form of the embedding function $F(\mathbf{x})$.

2.1. Approximating $\mathbf{b}_i$ by nonlinear embedding

In general, we can adopt any suitable embedding learning algorithms for $F(\mathbf{x})$, linear or nonlinear. Here we use the following simple yet powerful nonlinear form

$$F(\mathbf{x}) = \mathbf{P}^\top \phi(\mathbf{x}) \tag{4}$$

where $\phi(\mathbf{x})$ is a m-dimensional column vector obtained by the RBF kernel mapping: $\phi(\mathbf{x}) = [\exp(||\mathbf{x} - \mathbf{a}_1||^2/\sigma), \cdots, \exp(||\mathbf{x}-\mathbf{a}_m||^2/\sigma)]^\top$, where $\{\mathbf{a}_j\}_{j=1}^m$ are the randomly selected m anchor points from the training samples and σ is the kernel width. The matrix $\mathbf{P} \in \mathbb{R}^{m\times L}$ projects the mapped data $\phi(\mathbf{x})$ onto the low dimensional space. Similar formulations as equation (4) are widely used as the kernel hash function in, *e.g.*, BRE [13] and KSH [20].

F-Step If we fix $\mathbf{B}$ in problem (3), the projection matrix $\mathbf{P}$ can be easily computed by regression

$$\mathbf{P} = (\phi(\mathbf{X})\phi(\mathbf{X})^\top)^{-1}\phi(\mathbf{X})\mathbf{B}^\top. \tag{5}$$

Note that this step is independent of the loss function $L(\cdot)$.

[1] At the high level, our method here shares similarities with [23] in the sense that both introduce a set of auxiliary variables and apply the same type of regularization. However, convex semidefinite programming is considered in [23] and we are more interested in nonconvex integer programming.

2.2. Joint learning with ℓ_2 loss

The formulation (3) is flexible and we can choose any proper loss function $L(\cdot)$ for the classification model. One simple choice is the ℓ_2 loss, with which problem (3) writes

$$\min_{\mathbf{B},\mathbf{W},F} \sum_{i=1}^{n} ||\mathbf{y}_i - \mathbf{W}^\top \mathbf{b}_i||^2 + \lambda||\mathbf{W}||^2 + \nu \sum_{i=1}^{n} ||\mathbf{b}_i - F(\mathbf{x}_i)||^2. \quad \text{s.t. } \mathbf{b}_i \in \{-1,1\}^L. \tag{6}$$

That is

$$\min_{\mathbf{B},\mathbf{W},F} ||\mathbf{Y} - \mathbf{W}^\top \mathbf{B}||^2 + \lambda||\mathbf{W}||^2 + \nu||\mathbf{B} - F(\mathbf{X})||^2 \quad \text{s.t. } \mathbf{B} \in \{-1,1\}^{L\times n}. \tag{7}$$

G-Step For problem (7), by fixing $\mathbf{B}$, it is easy to solve $\mathbf{W}$ by the regularized least squares problem, which has a closed-form solution:

$$\mathbf{W} = (\mathbf{B}\mathbf{B}^\top + \lambda \mathbf{I})^{-1}\mathbf{B}\mathbf{Y}^\top. \tag{8}$$

B-Step It is challenging to solve for $\mathbf{B}$ due to the discrete constraints. With all variables but $\mathbf{B}$ fixed, we write problem (7) as

$$\min_{\mathbf{B}} \quad ||\mathbf{Y} - \mathbf{W}^\top \mathbf{B}||^2 + \nu||\mathbf{B} - F(\mathbf{X})||^2 \quad \text{s.t.} \quad \mathbf{B} \in \{-1,1\}^{L\times n}. \tag{9}$$

The above problem is NP hard. Here an important observation is that for problem (9) *a closed-form solution for one row of* $\mathbf{B}$ *can be achieved by fixing all the other rows*. It means that we can iteratively learn one bit at a time. To see this, let us rewrite (9):

$$\min_{\mathbf{B}} \quad ||\mathbf{Y}||^2 - 2\mathrm{Tr}(\mathbf{Y}^\top \mathbf{W}^\top \mathbf{B}) + ||\mathbf{W}^\top \mathbf{B}||^2 + \nu(||\mathbf{B}||^2 - 2\mathrm{Tr}(\mathbf{B}^\top F(\mathbf{X})) + ||F(\mathbf{X})||^2) \quad \text{s.t.} \quad \mathbf{B} \in \{-1,1\}^{L\times n}, \tag{10}$$

which is equivalent to

$$\min_{\mathbf{B}} \quad ||\mathbf{W}^\top \mathbf{B}||^2 - 2\mathrm{Tr}(\mathbf{B}^\top \mathbf{Q}) \quad \text{s.t.} \quad \mathbf{B} \in \{-1,1\}^{L\times n}. \tag{11}$$

where $\mathbf{Q} = \mathbf{W}\mathbf{Y} + \nu F(\mathbf{X})$ and $\mathrm{Tr}(\cdot)$ is the trace norm.

We choose to learn the binary codes $\mathbf{B}$ by the *discrete cyclic coordinate descent (DCC)* method. In other words, We learn $\mathbf{B}$ bit by bit. Let $\mathbf{z}^\top$ be the l^{th} row of $\mathbf{B}$, $l = 1, \cdots, L$ and $\mathbf{B}'$ the matrix of $\mathbf{B}$ excluding $\mathbf{z}$. Then $\mathbf{z}$ is one bit for all n samples. Similarly, let $\mathbf{q}^\top$ be the l^{th} row of $\mathbf{Q}$, $\mathbf{Q}'$ the matrix of $\mathbf{Q}$ excluding $\mathbf{q}$, $\mathbf{v}^\top$ the l^{th} row of $\mathbf{W}$ and $\mathbf{W}'$ the matrix of $\mathbf{W}$ excluding $\mathbf{v}$. Then we have

$$\begin{aligned} ||\mathbf{W}^\top \mathbf{B}||^2 &= \mathrm{Tr}(\mathbf{B}^\top \mathbf{W}\mathbf{W}^\top \mathbf{B}) \\ &= const + ||\mathbf{z}\mathbf{v}^\top||^2 + 2\mathbf{v}^\top \mathbf{W}'^\top \mathbf{B}'\mathbf{z} \\ &= const + 2\mathbf{v}^\top \mathbf{W}'^\top \mathbf{B}'\mathbf{z}. \end{aligned} \tag{12}$$

Here $||\mathbf{z}\mathbf{v}^\top||^2 = \mathrm{Tr}(\mathbf{v}\mathbf{z}^\top \mathbf{z}\mathbf{v}^\top) = n\mathbf{v}^\top \mathbf{v} = const$.

Similarly, we have

$$\mathrm{Tr}(\mathbf{B}^\top \mathbf{Q}) = const + \mathbf{q}^\top \mathbf{z}. \tag{13}$$

Putting equations (12), (13), and (11) altogether, we have the following problem w.r.t. $\mathbf{z}$:

$$\min_{\mathbf{z}} \quad (\mathbf{v}^\top \mathbf{W}'^\top \mathbf{B}' - \mathbf{q}^\top)\mathbf{z} \quad \text{s.t.} \quad \mathbf{z} \in \{-1,1\}^n. \tag{14}$$

This problem has the optimal solution

$$\mathbf{z} = \mathrm{sgn}(\mathbf{q} - \mathbf{B}'^\top \mathbf{W}'\mathbf{v}). \tag{15}$$

It is easy to see form (15) that, each bit $\mathbf{z}$ is computed based on the pre-learned $L-1$ bits $\mathbf{B}'$. This is expected because, one can iteratively update each bit till the procedure converges with a set of better codes $\mathbf{B}$. In our experiments, the whole L bits for $\mathbf{X}$ can be iteratively learned in tL times by (15), where usually $t = 2 \sim 5$.

Algorithm 1 Supervised Discrete Hashing (SDH)

Input: Training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$; code length L; number of anchor points m; maximum iteration number t; parameters λ and ν.

Output: Binary codes $\{\mathbf{b}_i\}_{i=1}^n \in \{-1,1\}^{L\times n}$; hash function $H(\mathbf{x}) = \mathrm{sgn}(F(\mathbf{x}))$.

1. Randomly select m samples $\{\mathbf{a}_j\}_{j=1}^m$ from the training data and get the mapped training data $\phi(\mathbf{x})$ via the RBF kernel function.
2. Initialize $\mathbf{b}_i$ as a $\{-1,1\}^L$ vector randomly, $\forall i$.
3. Loop until converge or reach maximum iterations:
   - **G-Step:** Calculate $\mathbf{W}$ using equation (8) or multi-class SVM.
   - **F-Step:** Compute $\mathbf{P}$ using (5) to form $F(\mathbf{x})$.
   - **B-Step:** For the ℓ_2 loss, iteratively learn $\{\mathbf{b}_i\}_{i=1}^n$ bit by bit using the DCC method with equation (15); for the hinge loss, compute $\mathbf{b}_i$ by (22).

2.3. Joint learning with hinge loss

The hinge loss is usually used in SVM as the cost function. With L the hinge loss, problem (3) can be formulated

as

$$\min_{\mathbf{B},\mathbf{W},F,\xi} \quad \lambda||\mathbf{W}||^2 + \sum_{i=1}^{n}\xi_i + \nu\sum_{i=1}^{n}||\mathbf{b}_i - F(\mathbf{x}_i)||^2 \quad (16)$$
$$\text{s.t.} \quad \forall i,k \quad \mathbf{w}_{c_i}^\top\mathbf{b}_i + y_{ki} - \mathbf{w}_k^\top\mathbf{b}_i \geq 1 - \xi_i,$$
$$\mathbf{b}_i \in \{-1,1\}^L.$$

where c_i is the label of $\mathbf{x}_i$ and $\xi_i \geq 0$ is the slack variable. It is not difficult to see that this formulation is the standard multi-class SVM [3] except that it is regularized by the approximation loss of learning codes. The multi-class Boosting method [28] also applies similar formulation.

G-Step With $\mathbf{B}$ fixed, the classification matrix $\mathbf{W}$ in problem (16) can be easily solved by the multi-class SVM solver, like LIBLINEAR [6].

B-Step With all variables but $\mathbf{b}_i$ fixed, problem (16) writes

$$\min_{\mathbf{b}_i} \quad ||\mathbf{b}_i - F(\mathbf{x}_i)||^2 \quad (17)$$
$$\text{s.t.} \quad \forall k \quad \mathbf{w}_{c_i}^\top\mathbf{b}_i + y_{ki} - \mathbf{w}_k^\top\mathbf{b}_i \geq 1 - \xi_i. \quad (18)$$
$$\mathbf{b}_i \in \{-1,1\}^L.$$

Constraints (18) can be written as

$$\forall k \quad \mathbf{w}^{(ki)\top}\mathbf{b}_i + y^{(ki)} \geq 0, \quad (19)$$
$$\mathbf{w}^{(ki)} = \mathbf{w}_{c_i} - \mathbf{w}_k,$$
$$y^{(ki)} = y_{ki} - 1 + \xi_i.$$

Take (19) into problem (17), we have

$$\min_{\mathbf{b}_i} \quad ||\mathbf{b}_i - F(\mathbf{x}_i)||^2 - \delta\sum_{k=1}^{C}(\mathbf{w}^{(ki)\top}\mathbf{b}_i + y^{(ki)}) \quad (20)$$
$$\text{s.t.} \quad \mathbf{b}_i \in \{-1,1\}^L.$$

Here the regularization parameter δ is fixed to $1/\nu$ in our experiments.

Problem (20) can be easily written to

$$\max_{\mathbf{b}_i} \quad \mathbf{b}_i^\top\big(F(\mathbf{x}_i) + \frac{\delta}{2}\sum_{k=1}^{C}\mathbf{w}^{(ki)}\big) \quad (21)$$
$$\text{s.t.} \quad \mathbf{b}_i \in \{-1,1\}^L,$$

which has the optimal solution

$$\mathbf{b}_i = \text{sgn}(F(\mathbf{x}_i) + \frac{\delta}{2}\sum_{k=1}^{C}\mathbf{w}^{(ki)}). \quad (22)$$

The proposed Supervised Discrete Hashing (SDH) method is summarized in Algorithm 1.

3. Experiments

In this section, extensive experiments are conducted to evaluate the proposed hashing method in both computational efficiency and retrieval performance. We test our method on four large-scale image datasets: CIFAR-10[2], MNIST[3], NUS-WIDE and ImageNet. All the data samples are normalized to have unit length. The proposed method is compared against several state-of-the-art supervised hashing methods including BRE [13], MLH [24], SSH [34], CCA-ITQ [10], KSH [20], FastHash [17] and unsupervised methods including PCA-ITQ [10], AGH [21] and IMH [29] with t-SNE [33]. We use the public codes and suggested parameters of these methods from the corresponding authors. For SDH, we empirically set λ to 1 and ν to 1e-5; the maximum iteration number t is set to 5. For AGH, IMH and SDH, we use randomly sampled 1,000 anchor points.

For CIFAR-10 and MNIST, we report the compared results in terms of both hash lookup (precision of Hamming radius 2) and Hamming ranking (mean of average precision, MAP). 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.

The MNIST dataset consists of $70,000$ images, each of 784 dimensions, of handwritten digits from '0' to '9'. As a subset of the well-known 80M tiny image collection [32], CIFAR-10 consists of 60,000 images which are manually labelled as 10 classes with $6,000$ samples for each class. We represent each image in this dataset by a GIST feature vector [26] of dimension 512. For MNIST and CIFAR-10, the whole dataset is split into a test set with $1,000$ samples and a training set with all remaining samples. We will describe the dataset of NUS-WIDE and ImageNet in the corresponding subsection.

3.1. Comparison between the ℓ_2 loss and hinge loss

In this experiment, we compare the two loss functions we have discussed for the linear classifier in the proposed SDH framework: the ℓ_2 loss and the hinge loss. The comparative results are reported in Table 1. It is interesting to see that these two loss functions achieve close results in both precision and MAP. In all the following experiments, we use the ℓ_2 loss for evaluation of our method.

3.2. Discrete or not?

To see how much the discrete optimization will benefit the hash code learning, in this subsection, we perform a comparison of our hash formulation (3) with or without the discrete constraints. The comparative results on CIFAR are shown in Table 2. As can be seen, our discrete hashing framework SDH consistently yields better hash codes than

[2] http://www.cs.toronto.edu/~kriz/cifar.html
[3] http://yann.lecun.com/exdb/mnist/

Table 1: Comparative results for the ℓ_2 loss and hinge loss used in the proposed SDH method.

	Loss	32 bits	64 bits	96 bits
Precision	ℓ_2	0.5090	0.4229	0.3515
	Hinge	0.5143	0.4328	0.3322
MAP	ℓ_2	0.4307	0.4555	0.4582
	Hinge	0.4449	0.4635	0.4581

Table 2: Comparative results of our method with discrete constraints or relaxation.

	Constraint	32 bits	64 bits	96 bits
Precision	Discrete	0.5090	0.4229	0.3515
	Relaxed	0.4718	0.3354	0.1685
MAP	Discrete	0.4307	0.4555	0.4582
	Relaxed	0.3777	0.4150	0.4244

the relaxed one by removing the sign function. In particular for precision, the performance gaps between these two methods are increased with longer hash bits.

3.3. Retrieval on tiny natural images

First let us test the impact of the number of anchor points used in our method on the retrieval performance. We take CIFAR-10 as the test database in this experiment. The results are shown in Table 3. All the results of SDH in Table 2 are average results based on 10 independent runs. We didn't report the standard deviations since they are very small (e.g., 0.0015 of Precision and 0.0026 of MAP with 59000 training images and 1000 anchors). We are not surprising to see that for our method, the more anchor points yield better performance in both precision and MAP. The training and testing time with varying anchor points are also shown. We can also see that higher computation cost will be taken with more anchors. We set the number of anchors to 1,000 in the following experiments for test efficiency.

From the last two columns in Table 3, we can see that the training time cost of our method is very low, which allow the method to be applied on the whole training data. The proposed method needs only about 1 minute to train on all the 59,000 training images. In contrast, BRE, MLH and FastHash needs from about 40 minutes to more than a few hours to train on only 5,000 samples. CCA-ITQ, SSH and all the three unsupervised hashing methods PCA-ITQ, AGH and IMH are also very efficient, however, they perform much worse than our method in both precision and MAP. For fair comparisons and evaluation efficiency, in the following experiments we use 5,000 training samples for all the methods unless otherwise specified.

We further show the precision and MAP scores of the compared methods with different code lengths in Figure 1. We can clearly see that the proposed SDH achieves the best precisions and MAPs for all code lengths. CCA-ITQ and FastHash perform well in Hamming distance 2 precision with short codes while outperformed by SSH and MLH with 96 bits. In terms of MAP, FastHash performs the second best, however it is still much worse than SDH.

3.4. MNIST: retrieval with hand-written digits

The comparative results on MNIST in precision and recall of Hamming distance 2 are reported in Figure 2. It is clear that our method outperforms all other methods with almost all code lengths in both precision and recall. For precision, BRE, MLH, SSH and CCA-ITQ suffer rapid performance deteriorations with increasing code lengths. FastHash and KSH achieve better results than these methods with more than 64 bits. However the performance of FastHash and KSH is still inferior to that of our method by large margins. As a matter of recall, our SDH consistently outperforms all other methods. Among all other compared algorithms, FastHash and KSH achieves the best results.

3.5. NUS-WIDE: retrieval with multiple labels

The NUS-WIDE[4] database contains about 270,000 images collected from Flickr. NUS-WIDE is associated with 81 ground truth concept labels, with each image containing multiple semantic labels. We define the true neighbors of a query as the images sharing at least one labels with the query image. The provided 500-dimensional Bag-of-Words features are used. As in [20], we collect the 21 most frequent label for test. For each label, 100 images are uniformly sampled for the query set and the remaining images are for the training set. For this large dataset, we use all the training samples for the efficient SDH and CCA-ITQ. For BRE, MLH and KSH, we sample 5,000 images for training.

The precisions and MAPs obtained by the compared methods with varying code lengths are shown in Figure 3. We can see from Figure 3 that with short code lengths CCA-ITQ achieves the best results in precision, however its performance deteriorates rapidly with increasing code lengths. The proposed SDH achieves achieves the highest precision when code length is longer than 32. In terms of MAP, SDH performs the best with almost all code lengths. KSH and CCA-ITQ also achieve promising results. The superior results of SDH demonstrate the effectiveness of our method on the retrieval task of data with multiple semantic labels.

3.6. ImageNet: retrieval with high dimensional features

As a subset of ImageNet [5], the large dataset ILSVRC 2012 contains over 1.2 million images of totally 1,000 categories. We use the provided training set as the retrieval database and 50,000 images from the validation set as the

[4] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

Table 3: Results in precision of Hamming distance within radius 2, MAP, training time and testing time on CIFAR-10. Results with 64 bits are reported. For our method, the number of anchors varies from 300 to 3000. For SSH, we use 5,000 labelled data points for similarity matrix construction. The training and testing time are in seconds. We The experiments are conducted on a desktop PC with an Intel quad-core 3.4GHZ CPU and 32G RAM.

Method	# training	# anchor	Precision	MAP	Training time	Test time
SDH	5000	300	0.3092	0.3400	10.3	1.7e-6
	5000	500	0.3494	0.3707	11.3	2.2e-6
	5000	1000	0.3585	0.4026	12.6	2.6e-6
	5000	3000	0.3780	0.4361	26.8	8.7e-6
	59000	1000	**0.4229**	**0.4555**	62.6	2.6e-6
BRE	5000	-	0.1299	0.1156	12042.0	6.4e-5
MLH	5000	-	0.2251	0.1730	2297.5	3.2e-5
KSH	5000	1000	0.1656	0.3822	2625.0	3.1e-6
SSH	59000	-	0.2860	0.2091	96.9	3.6e-6
CCA-ITQ	59000	-	0.3524	0.3379	4.3	1.7e-6
FastHash	59000	-	0.1880	0.4187	1340.7	7.1e-4
PCA-ITQ	59000	-	0.0160	0.1763	119.3	1.6e-6
AGH	59000	1000	0.3462	0.1513	67.3	3.9e-6
IMH	59000	1000	0.2879	0.2055	125.7	2.8e-6

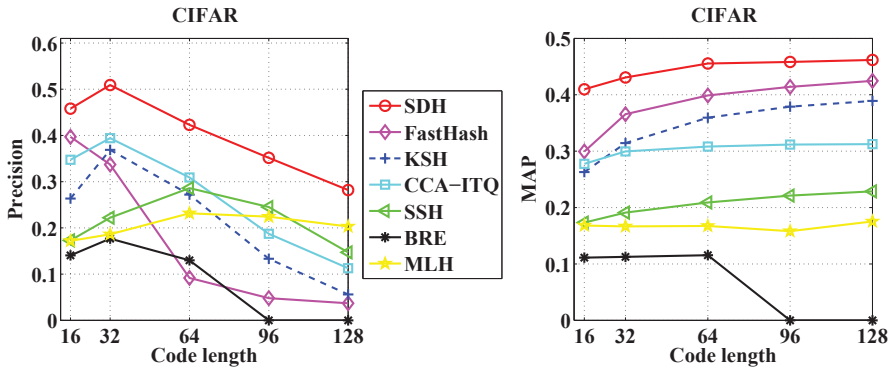

Figure 1: Results of the compared methods in precision of Hamming distance 2 and MAP on CIFAR-10 with code length from 16 to 128.

Table 4: Comparative results in precision of Hamming distance within radius 2 for various methods on the ILSVRC 2012 dataset. For this large dataset, 10,000 samples are used for training.

Method	64 bits	128 bits
BRE	0.3780	0.0235
SSH	0.0229	0.0005
CCA-ITQ	0.1786	0.0215
KSH	0.4850	0.2901
FastHash	0.3121	0.3564
SDH	**0.6495**	**0.5863**

query set. As in [17], we use the 4096-dimensional features extracted by the convolution neural networks (CNN) model. For this large dataset, we only report the compared precisions within Hamming radius 2 due to the high computational cost of MAP evaluation. The results are shown in Table 4. It is clear that on this large dataset the proposed method significantly outperforms all other approaches by even larger gaps. For precision with 64 bits, our method achieves a precision of 64.95%, which is higher than the second best result (obtained by KSH) by over 16%. With 128 bits, BRE, SSH and CCA-ITQ obtain very low precisions for the sparse data distribution in the Hamming space, while SDH, FasHash and KSH still achieve promising results.

3.7. Classification with binary codes

In this subsection, we use the CIFAR-10 dataset for example to test the performance of the learned binary codes by various hashing algorithms on classification problems. In this experiment, we compare SDH with several other supervised hashing methods including BRE, CCA-ITQ, SSH, KSH and FastHash. With the binary codes obtained

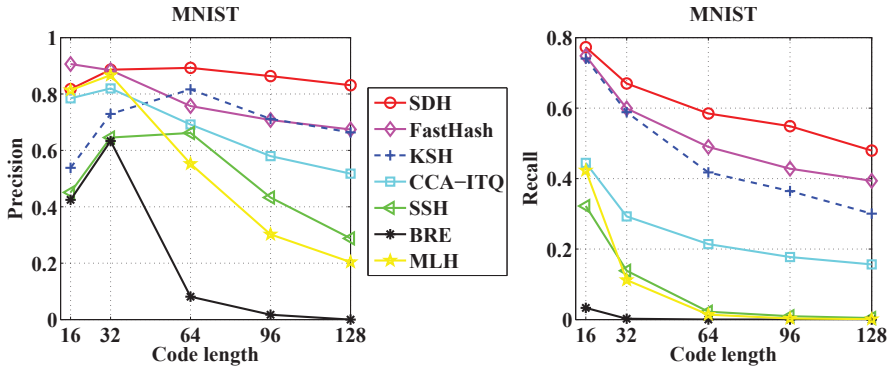

Figure 2: Results of the compared methods in precision and recall of Hamming distance 2 on MNIST with code length from 16 to 128.

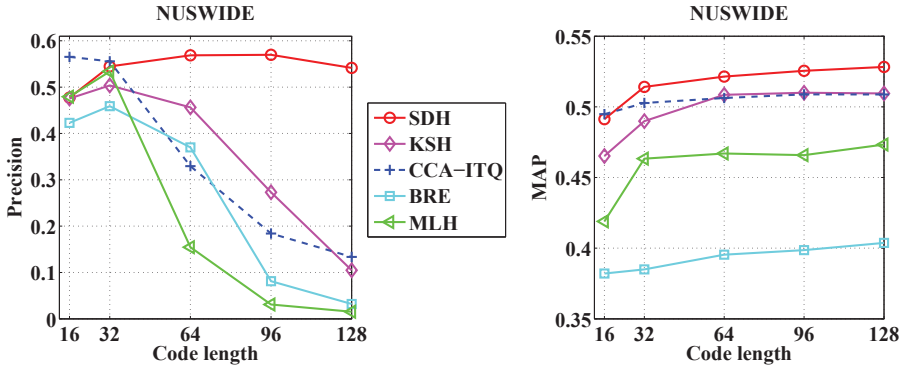

Figure 3: Compared precision and MAP results on NUS-WIDE with different code lengths.

Table 5: Classification accuracies on CIFAR-10 obtained by various hash codes. Linear SVM is used.

Method	64 bits	128 bits
BRE	0.332	0.335
CCA-ITQ	0.564	0.570
SSH	0.432	0.460
KSH	0.590	0.598
FastHash	0.578	0.603
SDH	**0.636**	**0.651**
GIST	0.580	

by these methods, we apply linear SVM for classification. The LIBLINEAR [6] solver is used. We report the results in Table 5. From Table 5, we can see that the proposed SDH achieves the best classification accuracies with both 64 bits and 128 bits, as SDH has explicitly considered the linear classification performance in its learning objective. We also test the 512-dim GIST features. We can see that the binary codes obtained by SDH (also FastHash and KSH) are even more discriminant than the continuous features. This is expected because the hash codes can be viewed as nonlinearly transformed feature vectors of the original GIST vectors. Hence linear classification on hash codes is equivalent to learn a nonlinear classifier on the original data.

4. Conclusions

In this paper, we revisited the supervised hashing problem. To leverage semantic label information, we formulated a joint learning objective which integrates hash code generation and linear classifier training. When optimizing this objective, the generated hash codes are expected to be optimal to the jointly trained classifier. By the design of the objective, we decomposed the supervised hashing problem into three sub-problems. To solve the most critical subproblem - discrete optimization for binary hash bits, we proposed a discrete cyclic coordinate descent (DCC) algorithm. Coupled with carefully chosen loss functions for the target classifier, the DCC algorithm yields the *optimal* hash bits in a *closed form*. As such, the entire optimization procedure remains very efficient and thereby enables our approach, called Supervised Discrete Hashing (SDH), to deal with practical massive data. The experimental results on four public image datasets demonstrated the efficacy of SDH for large-scale image retrieval. We further showed that the binary codes generated by SDH can also achieve promising classification performance.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[2] M. M. Bronstein and P. Fua. LDAHash: Improved matching with smaller descriptors. *IEEE TPAMI*, 34(1):66–78, 2012.

[3] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2002.

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. International Symposium on Computational Geometry*, 2004.

[5] 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.

[6] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[7] T. Ge, K. He, and J. Sun. Graph cuts for supervised binary coding. In *Proc. ECCV*, 2014.

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

[9] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *Proc. CVPR*, 2013.

[10] 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.

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

[12] W. Kong and W.-J. Li. Isotropic hashing. In *NIPS 25*, 2012.

[13] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS 22*, 2009.

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

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

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

[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.

[18] G. Lin, C. Shen, D. Suter, and A. v. d. Hengel. A general two-step approach to learning-based hashing. In *Proc. ICCV*, 2013.

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

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

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

[22] W. Liu, J. Wang, Y. Mu, S. Kumar, and S.-F. Chang. Compact hyperplane hashing with bilinear functions. In *Proc. ICML*, 2012.

[23] J. Malick, J. Povh, F. Rendl, and A. Wiegele. Regularization methods for semidefinite programming. *SIAM Journal on Optimization*, 20(1):336–356, 2009.

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

[25] M. Norouzi, D. M. Blei, and R. R. Salakhutdinov. Hamming distance metric learning. In *NIPS 25*, 2012.

[26] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[27] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS 22*, 2009.

[28] C. Shen and Z. Hao. A direct formulation for totally-corrective multi-class boosting. In *Proc. CVPR*, 2011.

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

[30] 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.

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

[32] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE TPAMI*, 30(11):1958–1970, 2008.

[33] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 9:2579–2605, 2008.

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

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

[36] W. Wang and M. A. Carreira-Perpinán. The role of dimensionality reduction in classification. In *Proc. AAAI*, 2014.

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

[38] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS 21*, 2008.

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

[40] 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.