Column Sampling based Discrete Supervised Hashing

Wang-Cheng Kang, Wu-Jun Li and **Zhi-Hua Zhou**
National Key Laboratory for Novel Software Technology
Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023
Department of Computer Science and Technology, Nanjing University, China
kwc.oliver@gmail.com, {liwujun,zhouzh}@nju.edu.cn

Abstract

By leveraging semantic (label) information, supervised hashing has demonstrated better accuracy than unsupervised hashing in many real applications. Because the hashing-code learning problem is essentially a discrete optimization problem which is hard to solve, most existing supervised hashing methods try to solve a relaxed continuous optimization problem by dropping the discrete constraints. However, these methods typically suffer from poor performance due to the errors caused by the relaxation. Some other methods try to directly solve the discrete optimization problem. However, they are typically time-consuming and unscalable. In this paper, we propose a novel method, called column sampling based discrete supervised hashing (COSDISH), to directly learn the discrete hashing code from semantic information. COSDISH is an iterative method, in each iteration of which several columns are sampled from the semantic similarity matrix and then the hashing code is decomposed into two parts which can be alternately optimized in a discrete way. Theoretical analysis shows that the learning (optimization) algorithm of COSDISH has a constant-approximation bound in each step of the alternating optimization procedure. Empirical results on datasets with semantic labels illustrate that COSDISH can outperform the state-of-the-art methods in real applications like image retrieval.

Introduction

Although different kinds of methods have been proposed for approximate nearest neighbor (ANN) search (Indyk and Motwani 1998), hashing has become one of the most popular candidates for ANN search because it can achieve better performance than other methods in real applications (Weiss, Torralba, and Fergus 2008; Kulis and Grauman 2009; Zhang et al. 2010; Zhang, Wang, and Si 2011; Zhang et al. 2012; Strecha et al. 2012; Zhen and Yeung 2012; Rastegari et al. 2013; Lin et al. 2013b; Xu et al. 2013; Jin et al. 2013; Zhu et al. 2013; Wang, Zhang, and Si 2013; Zhou, Ding, and Guo 2014; Yu et al. 2014).

There have appeared two main categories of hashing methods (Kong and Li 2012; Liu et al. 2012; Zhang et al. 2014): data-independent methods and data-dependent methods. Representative data-independent methods include locality-sensitive hashing (LSH) (Andoni and Indyk 2006) and its variants. The data-dependent hashing can be further divided into unsupervised hashing and supervised hashing methods (Liu et al. 2012; Zhang et al. 2014). Unsupervised hashing methods, such as iterative quantization (ITQ) (Gong and Lazebnik 2011), isotropic hashing (IsoHash) (Kong and Li 2012), discrete graph hashing (DGH) (Liu et al. 2014) and scalable graph hashing (SGH) (Jiang and Li 2015), only use the feature information of the data points for learning without using any semantic (label) information. On the contrary, supervised hashing methods try to leverage semantic (label) information for hashing function learning. Representative supervised hashing methods include sequential projection learning for hashing (SPLH) (Wang, Kumar, and Chang 2010b), minimal loss hashing (MLH) (Norouzi and Fleet 2011), supervised hashing with kernels (KSH) (Liu et al. 2012), two-step hashing (TSH) (Lin et al. 2013a), latent factor hashing (LFH) (Zhang et al. 2014), FastH (Lin et al. 2014), graph cuts coding (GCC) (Ge, He, and Sun 2014) and supervised discrete hashing (SDH) (Shen et al. 2015).

Supervised hashing has attracted more and more attention in recent years because it has demonstrated better accuracy than unsupervised hashing in many real applications. Because the hashing-code learning problem is essentially a discrete optimization problem which is hard to solve, most existing supervised hashing methods, such as KSH (Liu et al. 2012), try to solve a relaxed continuous optimization problem by dropping the discrete constraints. However, these methods typically suffer from poor performance due to the errors caused by the relaxation, which has been verified by the experiments in (Lin et al. 2014; Shen et al. 2015). Some other methods, such as FastH (Lin et al. 2014), try to directly solve the discrete optimization problem. However, they are typically time-consuming and unscalable. Hence, they have to sample only a small subset of the entire dataset for training even if a large-scale training set is given, which cannot achieve satisfactory performance in real applications.

In this paper, we propose a novel method, called column sampling based discrete supervised hashing (COSDISH), to directly learn the discrete hashing code from semantic information. The main contributions of COSDISH are listed as follows:

- COSDISH is iterative, and in each iteration column sampling (Zhang et al. 2014) is adopted to sample several columns from the semantic similarity matrix. Different from traditional sampling methods which try to sample only a small subset of the entire dataset for training, our column sampling method can exploit all the available data points for training.

- Based on the sampled columns, the hashing-code learning procedure can be decomposed into two parts which can be alternately optimized in a discrete way. The discrete optimization strategy can avoid the errors caused by relaxation in traditional continuous optimization methods.
- Theoretical analysis shows that the learning (optimization) algorithm has a constant-approximation bound in each step of the alternating optimization procedure.
- Empirical results on datasets with semantic labels illustrate that COSDISH can outperform the state-of-the-art methods in real applications, such as image retrieval.

Notation and Problem Definition

Notation

Boldface lowercase letters like $\mathbf{a}$ denote vectors, and the ith element of $\mathbf{a}$ is denoted as a_i. Boldface uppercase letters like $\mathbf{A}$ denote matrices. $\mathbf{I}$ denotes the identity matrix. $\mathbf{A}_{i*}$ and $\mathbf{A}_{*j}$ denote the ith row and the jth column of $\mathbf{A}$, respectively. A_{ij} denotes the element at the ith row and jth column in $\mathbf{A}$. $\mathbf{A}^{-1}$ denotes the inverse of $\mathbf{A}$, and $\mathbf{A}^T$ denotes the transpose of $\mathbf{A}$. $|\cdot|$ denotes the cardinality of a set, i.e., the number of elements in the set. $\|\cdot\|_F$ denotes the Frobenius norm of a vector or matrix, and $\|\cdot\|_1$ denotes the L_1 norm of a vector or matrix. $sgn(\cdot)$ is the element-wise sign function which returns 1 if the element is a positive number and returns -1 otherwise.

Problem Definition

Suppose we have n points $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ where $\mathbf{x}_i$ is the feature vector of point i. We can denote the feature vectors of the n points in a compact matrix form $\mathbf{X} \in \mathbb{R}^{n\times d}$, where $\mathbf{X}_{i*}^T = \mathbf{x}_i$. Besides the feature vectors, the training set of supervised hashing also contains a semantic similarity matrix $\mathbf{S} \in \{-1,0,1\}^{n\times n}$, where $S_{ij} = 1$ means that point i and point j are semantically similar, $S_{ij} = -1$ means that point i and point j are semantically dissimilar, and $S_{ij} = 0$ means that whether point i and point j are semantically similar or not is unknown. Here, the semantic information typically refers to semantic labels provided with manual effort. In this paper, we assume that $\mathbf{S}$ is fully observed without missing entries, i.e., $\mathbf{S} \in \{-1,1\}^{n\times n}$. This assumption is reasonable because in many cases we can always get the semantic label information between two points. Furthermore, our model in this paper can be easily adapted to the cases with missing entries.

The goal of supervised hashing is to learn a binary code matrix $\mathbf{B} \in \{-1,1\}^{n\times q}$, where $\mathbf{B}_{i*}$ denotes the q-bit code for training point i. Furthermore, the learned binary codes should preserve the semantic similarity in $\mathbf{S}$. Although the goal of supervised hashing can be formulated by different optimization problems, the commonly used one is as follows:

$$\min_{\mathbf{B}\in\{-1,1\}^{n\times q}} \|q\mathbf{S} - \mathbf{B}\mathbf{B}^T\|_F^2, \tag{1}$$

where $\|q\mathbf{S} - \mathbf{B}\mathbf{B}^T\|_F^2 = \sum_{i=1}^n \sum_{j=1}^n (qS_{ij} - \mathbf{B}_{i*}\mathbf{B}_{j*}^T)^2$.

The main idea of (1) is to adopt the inner product, which reflects the opposite of the Hamming distance, of two binary codes to approximate the similarity label with the square loss. This model has been widely used in many supervised hashing methods (Liu et al. 2012; Lin et al. 2013a; Zhang and Li 2014; Lin et al. 2014; Xia et al. 2014; Leng et al. 2014). LFH (Zhang et al. 2014) also uses the inner product to approximate the similarity label, but it uses the logistic loss rather than the square loss in (1).

Problem (1) is a discrete optimization problem which is hard to solve. Most existing methods optimize it by dropping the discrete constraint (Liu et al. 2012; Lin et al. 2013a). To the best of our knowledge, only one method, called FastH (Lin et al. 2014), has been proposed to directly solve the discrete optimization problem in (1). Due to the difficulty of discrete optimization, FastH adopts a bit-wise learning strategy which uses a Block Graph-Cut method to get the local optima and learn one bit at a time. The experiments of FastH show that FastH can achieve better accuracy than other supervised methods with continuous relaxation.

It is easy to see that both time complexity and storage complexity are $O(n^2)$ if all the supervised information in $\mathbf{S}$ is used for training. Hence, all the existing methods, including relaxation-based continuous optimization methods and discrete optimization methods, sample only a small subset with m ($m < n$) points for training where m is typically several thousand even if we are given a large-scale training set. Because some training points are discarded, all these existing methods cannot achieve satisfactory accuracy.

Therefore, to get satisfactory accuracy, we need to solve the problem in (1) from two aspects. On one hand, we need to adopt proper sampling strategy to effectively exploit all the n available points for training. On the other hand, we need to design strategies for discrete optimization. This motivates the work in this paper.

COSDISH

This section presents the details of our proposed method called COSDISH. More specifically, we try to solve the two aspects stated above, sampling and discrete optimization, for the supervised hashing problem.

Column Sampling

As stated above, both time complexity and storage complexity are $O(n^2)$ if all the supervised information in $\mathbf{S}$ is used for training. Hence, we have to perform sampling for training, which is actually adopted by almost all existing methods, such as KSH, TSH, FastH and LFH. However, all existing methods except LFH try to sample only a small subset with m ($m < n$) points for training and discard the rest training points, which leads to unsatisfactory accuracy.

The special case is LFH, which proposes to sample several columns from $\mathbf{S}$ in each iteration and several iterations are performed for training. In this paper, we adopt the same *column sampling* method as LFH (Zhang et al. 2014) for our COSDISH. Unlike LFH which adopts continuous relaxation for learning, in COSDISH we propose a novel discrete optimization (learning) method based on column sampling.

More specifically, in each iteration we randomly sample a subset Ω of $\mathcal{N} = \{1, 2, \ldots, n\}$ and then choose the semantic similarity between all n points and those points indexed by Ω. That is to say, we sample $|\Omega|$ columns of $\mathbf{S}$ with the column numbers being indexed by Ω. We use $\widetilde{\mathbf{S}} \in \{-1, 1\}^{n \times |\Omega|}$ to denote the sampled sub-matrix of similarity.

We can find that there exist two different kinds of points in each iteration, one being those indexed by Ω and the other being those indexed by $\Gamma = \mathcal{N} - \Omega$. We use $\widetilde{\mathbf{S}}^{\Omega} \in \{-1, 1\}^{|\Omega| \times |\Omega|}$ to denote a sub-matrix formed by the rows of $\widetilde{\mathbf{S}}$ indexed by Ω. $\widetilde{\mathbf{S}}^{\Gamma} \in \{-1, 1\}^{|\Gamma| \times |\Omega|}$, $\mathbf{B}^{\Omega} \in \{-1, 1\}^{|\Omega| \times q}$ and $\mathbf{B}^{\Gamma} \in \{-1, 1\}^{|\Gamma| \times q}$ are defined in a similar way.

According to the problem in (1), the problem associated with the sampled columns in each iteration can be reformulated as follows:

$$\min_{\mathbf{B}^{\Omega}, \mathbf{B}^{\Gamma}} \|q\widetilde{\mathbf{S}}^{\Gamma} - \mathbf{B}^{\Gamma}[\mathbf{B}^{\Omega}]^T\|_F^2 + \|q\widetilde{\mathbf{S}}^{\Omega} - \mathbf{B}^{\Omega}[\mathbf{B}^{\Omega}]^T\|_F^2. \quad (2)$$

Alternating Optimization

We propose an alternating optimization strategy, which contains several iterations with each iteration divided into two alternating steps, to solve the problem in (2). More specifically, we update $\mathbf{B}^{\Gamma}$ with $\mathbf{B}^{\Omega}$ fixed, and then update $\mathbf{B}^{\Omega}$ with $\mathbf{B}^{\Gamma}$ fixed. This two-step alternating optimization procedure will be repeated for several times.

Update $\mathbf{B}^{\Gamma}$ with $\mathbf{B}^{\Omega}$ Fixed When $\mathbf{B}^{\Omega}$ is fixed, the objective function of $\mathbf{B}^{\Gamma}$ is given by:

$$\underset{\mathbf{B}^{\Gamma} \in \{-1,1\}^{|\Gamma| \times q}}{f_2(\mathbf{B}^{\Gamma})} = \|q\widetilde{\mathbf{S}}^{\Gamma} - \mathbf{B}^{\Gamma}[\mathbf{B}^{\Omega}]^T\|_F^2.$$

Minimizing $f_2(\cdot)$ can be viewed as a discrete least square problem. By relaxing the $\mathbf{B}^{\Gamma}$ into continuous values, it is easy to get the optimal continuous solution. However, after we quantize the continuous solution into discrete solution, the error bound caused by quantization cannot be guaranteed. Hence, even if the continuous solution is optimal for the relaxed continuous problem, the discrete solution might be far away from the optimal solution of the original problem.

Here, we design a method to solve it in a discrete way with a constant-approximation bound. Let's consider the following problem $f_1(\cdot)$ which changes the loss from Frobenius norm in $f_2(\cdot)$ to L_1 norm:

$$\underset{\mathbf{B}^{\Gamma} \in \{-1,1\}^{|\Gamma| \times q}}{f_1(\mathbf{B}^{\Gamma})} = \|q\widetilde{\mathbf{S}}^{\Gamma} - \mathbf{B}^{\Gamma}[\mathbf{B}^{\Omega}]^T\|_1. \quad (3)$$

It's easy to find that when $\mathbf{B}^{\Gamma} = sgn(\widetilde{\mathbf{S}}^{\Gamma}\mathbf{B}^{\Omega})$, $f_1(\cdot)$ reaches its minimum. Furthermore, we have the following theorem.

Theorem 1. *Suppose that $f_1(\mathbf{F}_1^*)$ and $f_2(\mathbf{F}_2^*)$ reach their minimum at the points $\mathbf{F}_1^*$ and $\mathbf{F}_2^*$, respectively. We have $f_2(\mathbf{F}_1^*) \leq 2q f_2(\mathbf{F}_2^*)$.*

The proof of Theorem 1 can be found in the supplementary material[1].

That is to say, if we use the solution of $f_1(\cdot)$, i.e., $\mathbf{B}^{\Gamma} = \mathbf{F}_1^* = sgn(\widetilde{\mathbf{S}}^{\Gamma}\mathbf{B}^{\Omega})$, as the solution of $f_2(\cdot)$, the solution is a $2q$-approximation solution. Because q is usually small, we can say that it is a constant-approximation solution, which means that we can get an error bound for the original problem.

Because the elements of $\widetilde{\mathbf{S}}^{\Gamma}\mathbf{B}^{\Omega}$ can be zero, we set $\mathbf{B}^{\Gamma}_{(t)} = e\_sgn(\widetilde{\mathbf{S}}^{\Gamma}\mathbf{B}^{\Omega}, \mathbf{B}^{\Gamma}_{(t-1)})$ in practice:

$$e\_sgn(b1, b2) = \begin{cases} 1 & b1 > 0 \\ b2 & b1 = 0 \\ -1 & b1 < 0 \end{cases}$$

where t is the iteration number, and $e\_sgn(\cdot, \cdot)$ is applied in an element-wise manner.

Update $\mathbf{B}^{\Omega}$ with $\mathbf{B}^{\Gamma}$ Fixed When $\mathbf{B}^{\Gamma}$ is fixed, the sub-problem of $\mathbf{B}^{\Omega}$ is given by:

$$\min_{\mathbf{B}^{\Omega} \in \{-1,1\}^{|\Omega| \times q}} \|q\widetilde{\mathbf{S}}^{\Gamma} - \mathbf{B}^{\Gamma}[\mathbf{B}^{\Omega}]^T\|_F^2 + \|q\widetilde{\mathbf{S}}^{\Omega} - \mathbf{B}^{\Omega}[\mathbf{B}^{\Omega}]^T\|_F^2.$$

Inspired by TSH (Lin et al. 2013a), we can transform the above problem to q binary quadratic programming (BQP) problems. The optimization of the kth bit of $\mathbf{B}^{\Omega}$ is given by:

$$\min_{\mathbf{b}^k \in \{-1,1\}^{|\Omega|}} [\mathbf{b}^k]^T \mathbf{Q}^{(k)} \mathbf{b}^k + [\mathbf{b}^k]^T \mathbf{p}^{(k)} \quad (4)$$

where $\mathbf{b}^k$ denotes the kth column of $\mathbf{B}^{\Omega}$, and

$$\begin{aligned} Q^{(k)}_{\substack{i,j \\ i \neq j}} &= -2(q\widetilde{S}^{\Omega}_{i,j} - \sum_{m=1}^{k-1} b^m_i b^m_j), \; Q^{(k)}_{i,i} = 0, \\ p^{(k)}_i &= -2\sum_{l=1}^{|\Gamma|} B^{\Gamma}_{l,k}(q\widetilde{S}^{\Gamma}_{l,i} - \sum_{m=1}^{k-1} B^{\Gamma}_{l,m} B^{\Omega}_{i,m}). \end{aligned}$$

Note that the formulation in (4) is not the same as that in TSH due to the additional linear term. More details about the above derivation can be found in the supplementary material.

Then, we need to turn the problem in (4) into a standard BQP form in which the domain of binary variable is $\{0, 1\}$ and there are no linear terms.

First, we transform the domain of $\mathbf{b}^k$ from $\{-1, 1\}^{|\Omega|}$ to $\{0, 1\}^{|\Omega|}$. Let $\overline{\mathbf{b}}^k = \frac{1}{2}(\mathbf{b}^k + \mathbf{1})$, we have:

$$\begin{aligned} &[\mathbf{b}^k]^T \mathbf{Q}^{(k)} \mathbf{b}^k + [\mathbf{b}^k]^T \mathbf{p}^{(k)} \\ =&4\sum_{m=1}^{|\Omega|}\sum_{l=1}^{|\Omega|} \overline{b}^k_m \overline{b}^k_l Q^{(k)}_{m,l} + 2\sum_{m=1}^{|\Omega|} \overline{b}^k_m (p^{(k)}_m - \sum_{l=1}^{|\Omega|}(Q^{(k)}_{m,l} + Q^{(k)}_{l,m})) \\ &+ const, \end{aligned}$$

where *const* is a constant.

Hence, problem (4) can be reformulated as follows:

$$\min_{\overline{\mathbf{b}}^k \in \{0,1\}^{|\Omega|}} [\overline{\mathbf{b}}^k]^T \overline{\mathbf{Q}}^{(k)} \overline{\mathbf{b}}^k + [\overline{\mathbf{b}}^k]^T \overline{\mathbf{p}}^{(k)}, \quad (5)$$

[1]The supplementary material can be downloaded from `http://cs.nju.edu.cn/lwj/paper/COSDISH_sup.pdf`.

where $\overline{\mathbf{Q}}^{(k)} = 4\mathbf{Q}^{(k)}, \overline{p}_i^{(k)} = 2[p_i^{(k)} - \sum_{l=1}^{|\Omega|}(Q_{i,l}^{(k)} + Q_{l,i}^{(k)})]$. More details about the above derivation can be found in the supplementary material.

Furthermore, BQP can be turned into an equivalent form without linear terms (Yang 2013). That is to say, the problem in (5) can be rewritten as follows:

$$\begin{aligned} \min \quad & [\widetilde{\mathbf{b}}^k]^T \widetilde{\mathbf{Q}}^{(k)} \widetilde{\mathbf{b}}^k \\ s.t. \quad & \widetilde{\mathbf{b}}^k \in \{0,1\}^{|\Omega|+1}, \;\; \widetilde{b}^k_{|\Omega|+1} = 1 \end{aligned} \tag{6}$$

where

$$\widetilde{\mathbf{b}}^k = \begin{pmatrix} \overline{\mathbf{b}}^k \\ 1 \end{pmatrix}, \qquad \widetilde{\mathbf{Q}}^{(k)} = \begin{pmatrix} \overline{\mathbf{Q}}^{(k)} & \frac{1}{2}\overline{\mathbf{p}}^{(k)} \\ \frac{1}{2}[\overline{\mathbf{p}}^{(k)}]^T & 0 \end{pmatrix}.$$

A method proposed by (Yang 2013) can solve the problem in (6) with an additional constraint $\sum_{i=1}^{|\Omega|+1} \widetilde{b}_i^k = \lceil (|\Omega| + 1)/2 \rceil$. We also add this constraint to our problem to get a balanced result for each bit which has been widely used in hashing (Liu et al. 2014). So we reformulate problem (6) with the constraint as follows:

$$\begin{aligned} \min \quad & [\widetilde{\mathbf{b}}^k]^T \widetilde{\mathbf{Q}}^{(k)} \widetilde{\mathbf{b}}^k \\ s.t. \quad & \widetilde{\mathbf{b}}^k \in \{0,1\}^{M}, \quad \widetilde{b}^k_{M} = 1 \\ & \sum_{i=1}^{M} \widetilde{b}_i^k = H \end{aligned} \tag{7}$$

where $M = |\Omega| + 1$, and $H = \lceil M/2 \rceil$.

As in (Yang 2013), we transform the problem in (7) to an equivalent clustering problem: given a dataset $\mathcal{U} = \{\mathbf{u}_i \in \mathbb{R}^M\}_{i=1}^M$, we want to find a subset $\mathcal{U}'$ of size H that the sum of square of the distances within the subset $\mathcal{U}'$ is minimized. It can be formulated as:

$$\begin{aligned} \min \quad & \sum_{\mathbf{u}\in\mathcal{U}'} \|\mathbf{u} - \frac{1}{H}\sum_{\mathbf{v}\in\mathcal{U}'} \mathbf{v}\|^2 \\ s.t. \quad & \mathcal{U}' \subseteq \mathcal{U}, \quad |\mathcal{U}'| = H, \quad \mathbf{u}_M \in \mathcal{U}' \end{aligned} \tag{8}$$

Then, we have the following theorems.

Theorem 2. *Let us use a matrix $\mathbf{U}$ of size $M \times M$ to denote the dateset $\mathcal{U}$ with $\mathbf{U}_{*i} = \mathbf{u}_i$, and if $\mathbf{U}^T\mathbf{U} = \lambda\mathbf{I} - \widetilde{\mathbf{Q}}^{(k)} \succ 0$, then (7) and (8) are equivalent.*

Proof. We can use similar method in (Yang 2013) to prove it. □

According to (Yang 2013), we can always find such $\mathbf{U}$ and λ to satisfy $\lambda\mathbf{I} - \widetilde{\mathbf{Q}}^{(k)} \succ 0$ in Theorem 2. In practice, we can take a sufficiently large number as λ, and perform Cholesky decomposition on $\lambda\mathbf{I} - \widetilde{\mathbf{Q}}^{(k)}$ to get $\mathbf{U}$.

Theorem 3. *Assuming $\mathcal{U}'_*$ is the global solution of (8) and $f(\mathcal{U}') = \sum_{\mathbf{u}\in\mathcal{U}'} \|\mathbf{u} - \frac{1}{H}\sum_{\mathbf{v}\in\mathcal{U}'}\mathbf{v}\|^2$ is the objective function of (8), there exists an algorithm which can find a solution $\mathcal{U}'_1$ where*

$$f(\mathcal{U}'_1) \leq 2f(\mathcal{U}'_*).$$

That is to say, there exists a 2-approximation algorithm for (8).

Proof. Please refer to (Yang 2013) for the proof and algorithm. □

The Whole Algorithm The whole learning (optimization) algorithm is summarized in Algorithm 1.

Algorithm 1 Discrete optimization in COSDISH

Input: $\mathbf{S} \in \{-1,1\}^{n\times n}$, q, T_{sto}, T_{alt}, $|\Omega|$
Initialize binary code $\mathbf{B}$ by randomization
for $iter = 1 \to T_{sto}$ **do**
  Sample $|\Omega|$ columns of $\mathbf{S}$ to get $\widetilde{\mathbf{S}}$.
  Let Ω be the set of sampled column indices and $\Gamma = \mathcal{N} - \Omega$, with $\mathcal{N} = \{1, 2, \ldots n\}$.
  Split $\widetilde{\mathbf{S}}$ into $\widetilde{\mathbf{S}}^\Omega$ and $\widetilde{\mathbf{S}}^\Gamma$.
  Split $\mathbf{B}$ into $\mathbf{B}_{(0)}^\Omega$ and $\mathbf{B}_{(0)}^\Gamma$.
  for $t = 1 \to T_{alt}$ **do**
    for $k = 1 \to q$ **do**
      Construct problem (4) from $\mathbf{B}_{(t-1)}^\Gamma$, $\widetilde{\mathbf{S}}^\Omega$, $\widetilde{\mathbf{S}}^\Gamma$ and the first $k-1$ columns of $\mathbf{B}_{(t)}^\Omega$.
      Construct problem (5) from problem (4).
      Construct problem (7) from problems (5) and (6).
      Construct problem (8) from problem (7) by performing Cholesky decomposition.
      Using the 2-approximation algorithm (Yang 2013) to solve problem (8) and acquire the kth column of $\mathbf{B}_{(t)}^\Omega$.
    end for
    $\mathbf{B}_{(t)}^\Gamma = e\_sgn(\widetilde{\mathbf{S}}^\Gamma \mathbf{B}_{(t)}^\Omega, \mathbf{B}_{(t-1)}^\Gamma)$.
  end for
  Recover $\mathbf{B}$ by combining $\mathbf{B}_{(t)}^\Omega$ and $\mathbf{B}_{(t)}^\Gamma$.
end for
Output: $\mathbf{B} \in \{-1,1\}^{n\times q}$

In general, $10 \leq T_{sto} \leq 20$, $3 \leq T_{alt} \leq 10$ and $|\Omega| \geq q$ is enough to get satisfactory performance. Unless otherwise stated, we set $T_{sto} = 10$, $T_{alt} = 3$ and $|\Omega| = q$ in our experiments. Furthermore, in our experiments we find that our algorithm is not sensitive to the initialization. Hence, we adopt random initialization in this paper.

Remark 1. *Unlike other hashing methods such as TSH (Lin et al. 2013a) which try to solve the whole problem as q BQP problems, our alternating optimization strategy decomposes the whole problem into two sub-problems. Typically, $|\Omega| \ll |\Gamma|$, i.e., the number of variables in $\mathbf{B}^\Omega$ is far less than that in $\mathbf{B}^\Gamma$. Furthermore, the cost to get a solution for problem (3) is much lower than that to get a solution for BQP. Hence, the key idea of our alternating optimization strategy is to adopt a faster solution for the larger sub-problem, which makes our strategy much faster than TSH. Moreover, the faster solution of our strategy can also guarantee accuracy, which will be verified in the experiments.*

TSH adopts LBFGSB to solve the BQP problem in a continuous-relaxation way. We found that if LBFGSB is used to solve the BQP problem in our method (i.e., problem (6)), the accuracy of our method will be dramatically deteriorated. The reason is that the the solution of LBFGSB will hurt the quality of the solution of $\mathbf{B}^\Gamma$.

In addition, the graph-cut method used in FastH cannot be adapted to solve our problem (6), because problem (6) doesn't satisfy the sub-modular property required by the graph-cut method.

Soft Constraints

As mentioned in (Leng et al. 2014), when we can only get a subset of the semantic information, pushing two dissimilar points to have maximum Hamming distance may lead to over-fitting and unexpected result. Moreover, the number of dissimilar labels is typically far more than that of similar labels. Hence, we can also view it as a class-imbalance problem between positive and negative labels. Inspired by (Leng et al. 2014), we change the element -1 in our similarity matrix $\widetilde{\mathbf{S}}$ to a real value $0 < \beta < 1$. More specifically, we take $\beta = \frac{the\ number\ of\ 1\ in\ \widetilde{\mathbf{S}}}{the\ number\ of\ -1\ in\ \widetilde{\mathbf{S}}}$ empirically.

Please note that although soft constraints can further improve performance, the superior performance of our method mainly comes from the learning procedure rather than the soft constraints. Empirical verification about this can be found in the supplementary material.

Out-of-Sample Extension

Many supervised hashing methods can be viewed as two-step methods (Lin et al. 2013a): learn binary code in the first step, and then train q binary classifiers based on the feature matrix $\mathbf{X}$ and the learned code matrix $\mathbf{B}$ in the second step with each bit corresponding to one classifier (Zhang et al. 2014; Lin et al. 2013a; 2014; Xia et al. 2014). Besides those methods using linear classifiers (Wang, Kumar, and Chang 2010a; Zhang et al. 2014), some other methods use more powerful nonlinear classifiers, such as SVM with RBF kernel (Lin et al. 2013a), deep convolutional network (Xia et al. 2014) and boosted decision trees (Lin et al. 2014) and so on. In general, the more powerful classifiers we use for out-of-sample extension, the better accuracy we can achieve and also the more training time will be consumed (Lin et al. 2013a; 2014). FastH (Lin et al. 2014) adopts an efficient implementation of boosted decision trees for out-of-sample extension, which shows better accuracy and less training time than other methods like KSH and TSH with nonlinear classifiers.

Our COSDISH is also a two-step method. For out-of-sample extension, COSDISH chooses linear classifier and boosted decision trees in FastH to get two different variants. We will empirically evaluate these two variants in our experiments.

Complexity Analysis

The time complexity to construct problem (4) is $O(|\Gamma| \times |\Omega| + |\Omega|^2)$. Both the time complexity to construct problem (5) and that for problem (7) are $O(|\Omega|^2)$. Performing Cholesky decomposition on $\lambda\mathbf{I} - \widetilde{\mathbf{Q}}^{(k)}$ need $O(|\Omega|^3)$. The 2-approximation algorithm to solve the clustering problem need $O(|\Omega|^3 + |\Omega|^2 \log|\Omega|)$. For the $\mathbf{B}^{\Omega}$-subproblem, we need to solve q BQP problems. Hence, the complexity of $\mathbf{B}^{\Omega}$-subproblem is $O(q \times (|\Gamma| \times |\Omega| + |\Omega|^3))$. In addition, the time complexity of $\mathbf{B}^{\Gamma}$-subproblem is $O(q \times |\Gamma| \times |\Omega|)$. Therefore, the total time complexity is $O(T_{sto} \times T_{alt} \times q \times (|\Gamma| \times |\Omega| + |\Omega|^3))$, and the space complexity is $O(|\Gamma| \times |\Omega| + |\Omega|^2)$. If we take $|\Omega| = q$, then time complexity is $O(T_{sto} \times T_{alt} \times (nq^2 + q^4))$, which is linear to n. Typically, q is very small, e.g., less than 64. Hence, our method is scalable.

Experiment

We use real datasets to evaluate the effectiveness of our method. All the experiments are conducted on a workstation with 6 Intel Xeon CPU cores and 48GB RAM.

Dataset

Two image datasets with semantic labels are used to evaluate our method and the other baselines. They are CIFAR-10 (Krizhevsky 2009) and NUS-WIDE (Chua et al. 2009). Both of them have been widely used for hashing evaluation (Lin et al. 2013a; Zhang et al. 2014). Each instance in CIFAR-10 has a single label, and each instance in NUS-WIDE might have multi-labels.

CIFAR-10 contains 60,000 images. Each image is represented by a 512-dimension GIST feature vector extracted from the original color image of size 32×32. Each image is manually labeled to be one of the ten classes. Two images are considered to be semantically similar if they share the same class label. Otherwise, they are treated as semantically dissimilar.

NUS-WIDE includes 269,648 images crawled from Flickr with 81 ground-truth labels (tags). Each image is represented by a 1134-dimension feature vector after feature extraction. Each image might be associated with multi-labels. There also exist some images without any label, which are not suitable for our evaluation. After removing those images without any label, we get 209,347 images for our experiment. We consider two images to be semantically similar if they share at least one common label. Otherwise, they are semantically dissimilar.

For all the datasets, we perform normalization on feature vectors to make each dimension have zero mean and equal variance.

Experimental Settings and Baselines

As in LFH (Zhang et al. 2014), for all datasets we randomly choose 1000 points as validation set and 1000 points as query (test) set, with the rest of the points as training set. All experimental results are the average values of 10 independent random partitions.

Unless otherwise stated, COSDISH refers to the variant with soft constraints because in most cases it will outperform the variant without soft constraints (refer to the supplementary material). We use COSDISH to denote our method with linear classifier for out-of-sample extension, and COSDISH_BT to denote our method with boosted decision trees for out-of-sample extension.

Because existing methods (Lin et al. 2013a; Zhang et al. 2014) have shown that supervised methods can outperform unsupervised methods, we only compare our method with some representative supervised hashing methods, including SPLH (Wang, Kumar, and Chang 2010b), KSH (Liu et al. 2012), TSH (Lin et al. 2013a), LFH (Zhang et al. 2014), FastH (Lin et al. 2014) and SDH (Shen et al. 2015).

Table 1: Accuracy in terms of MAP. The best MAPs for each category are shown in boldface.

Method	CIFAR-10				NUS-WIDE			
	8-bits	16-bits	32-bits	64-bits	8-bits	16-bits	32-bits	64-bits
COSDISH	**0.4986**	**0.5768**	**0.6191**	**0.6371**	**0.5454**	**0.5940**	**0.6218**	**0.6329**
SDH	0.2642	0.3994	0.4145	0.4346	0.4739	0.4674	0.4908	0.4944
LFH	0.2908	0.4098	0.5446	0.6182	0.5437	0.5929	0.6025	0.6136
TSH	0.2365	0.3080	0.3455	0.3663	0.4593	0.4784	0.4857	0.4955
KSH	0.2334	0.2662	0.2923	0.3128	0.4275	0.4546	0.4645	0.4688
SPLH	0.1588	0.1635	0.1701	0.1730	0.3769	0.4077	0.4147	0.4071
COSDISH_BT	**0.5856**	**0.6681**	**0.7079**	**0.7346**	**0.5819**	**0.6316**	**0.6618**	**0.6786**
FastH	0.4230	0.5216	0.5970	0.6446	0.5014	0.5296	0.5541	0.5736

All the baselines are implemented by the source code provided by the corresponding authors. For LFH, we use the stochastic learning version with 50 iterations and each iteration sample q columns of the semantic similarity matrix as in (Zhang et al. 2014). For SPLH, KSH and TSH, we cannot use the entire training set for training due to high time complexity. As in (Liu et al. 2012; Lin et al. 2013a), we randomly sample 2000 points as training set for CIFAR-10 and 5000 points for NUS-WIDE. TSH can use different loss functions for training. For fair comparison, the loss function in KSH which is also the same as our COSDISH is used for training TSH. The SVM with RBF-kernel is used for out-of-sample-extension in TSH. For KSH, the number of support vectors is 300 for CIFAR-10, 1000 for NUS-WIDE. For FastH, boosted decision trees are used for out-of-sample extension. We use the entire training set for FastH training on CIFAR-10, and randomly sample 100,000 points for FastH training on NUS-WIDE. All the other hyperparameters and initialization strategy are the same as those suggested by the authors of the methods. In our experiment, we choose $|\Omega| = q$ for COSDISH which is the same as that in LFH. Actually, our method is not sensitive to $|\Omega|$.

Accuracy

The mean average precision (MAP) is a widely used metric for evaluating the accuracy of hashing (Zhang et al. 2014; Lin et al. 2014). Table 1 shows the MAP of our method and baselines. The eight methods in Table 1 can be divided into two different categories. The first category contains the first (top) six methods in Table 1 which use relatively weak classifiers for out-of-sample extension, and the second category contains the last (bottom) two methods in Table 1 which use a relatively strong classifier (i.e., boosted decision trees) for out-of-sample extension.

By comparing COSDISH to SPLH, KSH, TSH, LFH, SDH and FastH, we can find that COSDISH can outperform the other baselines in most cases. By comparing COSDISH_BT to COSDISH, we can find that the boosted decision trees can achieve much better accuracy than linear classifier for out-of-sample extension. By comparing COSDISH_BT to FastH which also uses boosted decision trees for out-of-sample extension, we can find that COSDISH_BT can outperform FastH, which verifies the effectiveness of our discrete optimization and column sampling strategies. In sum, our COSDISH and COSDISH_BT can achieve the state-of-the-art accuracy.

Scalability

We sample different numbers of training points from NUS-WIDE as the training set, and evaluate the scalability of COSDISH and baselines by assuming that all methods should use all the sampled training points for learning. Table 2 reports the training time, where the symbol '-' means that we can't finish the experiment due to out-of-memory errors. One can easily see that COSDISH, COSDISH_BT, SDH and LFH can easily scale to dataset of size 200,000 (200K) or even larger, while other methods either exceed the memory limit or consume too much time. LFH is more scalable than our COSDISH. However, as stated above, our COSDISH can achieve better accuracy than LFH with slightly increased time complexity. Hence, COSDISH is more practical than LFH for supervised hashing.

Table 2: Training time (in second) on subsets of NUS-WIDE

Method	3K	10K	50K	100K	200K
COSDISH	5.6	8.0	33.7	67.7	162.2
SDH	3.9	11.8	66.2	126.9	248.2
LFH	14.3	16.3	27.8	40.8	85.9
TSH	922.2	27360	$>$50000	-	-
KSH	1104	4446	$>$50000	-	-
SPLH	25.3	185	-	-	-
COSDISH_BT	60.2	69.1	228.3	422.6	893.3
FastH	172.3	291.6	1451	3602	-

Conclusion

In this paper, we have proposed a novel model called COSDISH for supervised hashing. COSDISH can directly learn discrete hashing code from semantic labels. Experiments on several datasets show that COSDISH can outperform other state-of-the-art methods in real applications.

Acknowledgements

This work is supported by the NSFC (No. 61472182, 61333014), and the Fundamental Research Funds for the Central Universities (No. 20620140510).

References

Andoni, A., and Indyk, P. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the Annual Symposium on Foundations of Computer Science*.

Chua, T.-S.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; and Zheng, Y. 2009. NUS-WIDE: A real-world web image database from national university of singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*.

Ge, T.; He, K.; and Sun, J. 2014. Graph cuts for supervised binary coding. In *Proceedings of the European Conference on Computer Vision*.

Gong, Y., and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Annual ACM Symposium on Theory of Computing*.

Jiang, Q.-Y., and Li, W.-J. 2015. Scalable graph hashing with feature transformation. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Jin, Z.; Hu, Y.; Lin, Y.; Zhang, D.; Lin, S.; Cai, D.; and Li, X. 2013. Complementary projection hashing. In *Proceedings of the IEEE International Conference on Computer Vision*.

Kong, W., and Li, W.-J. 2012. Isotropic hashing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto.

Kulis, B., and Grauman, K. 2009. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of the IEEE International Conference on Computer Vision*.

Leng, C.; Cheng, J.; Wu, J.; Zhang, X.; and Lu, H. 2014. Supervised hashing with soft constraints. In *Proceedings of the ACM International Conference on Conference on Information and Knowledge Management*.

Lin, G.; Shen, C.; Suter, D.; and Hengel, A. v. d. 2013a. A general two-step approach to learning-based hashing. In *Proceedings of the IEEE International Conference on Computer Vision*.

Lin, Y.; Jin, R.; Cai, D.; Yan, S.; and Li, X. 2013b. Compressed hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lin, G.; Shen, C.; Shi, Q.; van den Hengel, A.; and Suter, D. 2014. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.-G.; and Chang, S.-F. 2012. Supervised hashing with kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, W.; Mu, C.; Kumar, S.; and Chang, S. 2014. Discrete graph hashing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Norouzi, M., and Fleet, D. J. 2011. Minimal loss hashing for compact binary codes. In *Proceedings of the International Conference on Machine Learning*.

Rastegari, M.; Choi, J.; Fakhraei, S.; Hal, D.; and Davis, L. S. 2013. Predictable dual-view hashing. In *Proceedings of the International Conference on Machine Learning*.

Shen, F.; Shen, C.; Liu, W.; and Shen, H. T. 2015. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Strecha, C.; Bronstein, A. A.; Bronstein, M. M.; and Fua, P. 2012. Ldahash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(1):66–78.

Wang, J.; Kumar, O.; and Chang, S.-F. 2010a. Semi-supervised hashing for scalable image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, J.; Kumar, S.; and Chang, S.-F. 2010b. Sequential projection learning for hashing with compact codes. In *Proceedings of the International Conference on Machine Learning*.

Wang, Q.; Zhang, D.; and Si, L. 2013. Semantic hashing using tags and topic modeling. In *Proceedings of the International ACM SIGIR conference on research and development in Information Retrieval*.

Weiss, Y.; Torralba, A.; and Fergus, R. 2008. Spectral hashing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Xia, R.; Pan, Y.; Lai, H.; Liu, C.; and Yan, S. 2014. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Xu, B.; Bu, J.; Lin, Y.; Chen, C.; He, X.; and Cai, D. 2013. Harmonious hashing. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Yang, R. 2013. *New Results on Some Quadratic Programming Problems*. Phd thesis, University of Illinois at Urbana-Champaign.

Yu, Z.; Wu, F.; Yang, Y.; Tian, Q.; Luo, J.; and Zhuang, Y. 2014. Discriminative coupled dictionary hashing for fast cross-media retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhang, D., and Li, W.-J. 2014. Large-scale supervised multimodal hashing with semantic correlation maximization. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, D.; Wang, J.; Cai, D.; and Lu, J. 2010. Self-taught hashing for fast similarity search. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhang, Q.; Wu, Y.; Ding, Z.; and Huang, X. 2012. Learning hash codes for efficient content reuse detection. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhang, P.; Zhang, W.; Li, W.-J.; and Guo, M. 2014. Supervised hashing with latent factor models. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhang, D.; Wang, F.; and Si, L. 2011. Composite hashing with multiple information sources. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhen, Y., and Yeung, D.-Y. 2012. A probabilistic model for multimodal hash function learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Zhou, J.; Ding, G.; and Guo, Y. 2014. Latent semantic sparse hashing for cross-modal similarity search. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhu, X.; Huang, Z.; Shen, H. T.; and Zhao, X. 2013. Linear cross-modal hashing for efficient multimedia search. In *Proceedings of the ACM International Conference on Multimedia*.