A penalty function semi-continuous thresholding methods for image hash coding problems

Qian Chen[a], Zhengwei Shen[a], Zhe Chen[a]

[a]*School of Mathematics and Physics, University of Science and Technology Beijing, Beijing 100083, China.*

Abstract

Hashing process is an effective tool for handling large-scale data (for example images, videos or multi-model data) retrieval problems. To obtain better retrieval accuracy, hashing models usually are imposed with three rigorous constraints, i.e., discrete binary constraint, uncorrelated condition and the balanced constraint, which will lead the models to be NP-hard. In this study, we divide the whole constraints set into the uncorrelated (orthogonality) constraint and the binary discrete balance constraint, and propose a fast and accurate penalty function semi-continuous thresholding (PFSCT) hash coding algorithm based on forward-backward algorithms. In addition, we theoretically analyze the equivalence between the relaxed model and the original problems. Extensive numerical experiments on diverse large-scale benchmark datasets demonstrate comparable performance and effectiveness of the proposed method.

Keywords: Hash coding, image retrieval, orthogonality constraint, quantization error reduction.

1. Introduction

As an efficient dimensionality reducing and information retrieval technique to high-dimensional data, hash coding has been applied to a variety of large-scale information processing and machine learning problems including similarity

☆Fully documented templates are available in the elsarticle package on CTAN.
Email address: `ijwmip@ustb.edu.cn` (Zhengwei Shen)

search [1], image and video retrieval [2], objective detection [3], large scale multi-task learning [4], and recommendation system [5]. The task of hash coding is to achieve a p-bit binary vector $\mathbf{b}_i \in \{-1, 1\}^p$ for every sample $\mathbf{x}_i, (i = 1, 2, ..., n)$ of the given dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathsf{R}^{k\times n}$. Specifically, hash coding is to minimize a constrained optimization problem

$$\begin{aligned} &\min_{\mathbf{B}} \mathcal{F}(\mathbf{B}) \\ \text{s.t. } &\mathbf{B}\mathbf{B}^\top = \mathbf{I}_p, \mathbf{B}\mathbf{1}_n = \mathbf{0}_p, \mathbf{B} \in \{-1, 1\}^{p\times n} \end{aligned}, \tag{1}$$

where $\mathbf{B} \in \{-1, 1\}^{p\times n}$ is a binary code matrix to $\mathbf{X}$, $\mathbf{I}_p$ is a $p \times p$ identity matrix, $\mathbf{1}_n$ denotes n dimensional vectors with every element is 1, $\mathbf{0}_p$ denotes n dimensional vectors with every element is 0. Objective function $\mathcal{F}(\mathbf{B})$ in (1) is used to characterize certain weighted average Hamming distance and the associated regularizer. For example, in the unsupervised spectral graph hash model [6], we have

$$\mathcal{F}(\mathbf{B}) = \frac{1}{2} tr(\mathbf{B}\mathbf{L}\mathbf{B}^\top), \tag{2}$$

where $\mathbf{L}$ is the Laplacian matrix to dataset, p is the length of hash code of each data, n is the number of given data; but for the supervised SCDH model being established in [7], we have

$$\mathcal{F}(\mathbf{B}, \mathbf{A}, \mathbf{Z}) = ||r \cdot \mathbf{S} - \mathbf{B}\mathbf{Z}^\top||_F^2 + \lambda||\mathcal{K}_{\mathcal{Q}}\mathbf{A} - \mathbf{B}||_F^2 + \frac{\mu}{2}||\mathbf{B} - \mathbf{Z}||_F^2 + \gamma||\mathbf{A}||_F^2, \tag{3}$$

where $\mathbf{S} \in [-1, 1]^{n\times n}$ is the similarity matrix calculated by sample label vector, $\mathbf{Z}$ is auxiliary alias of $\mathbf{B} \in \{-1, 1\}^{n\times r}$. In addition, $\mathcal{K}_{\mathcal{Q}} \in \mathsf{R}^{n\times \mathcal{Q}}$ denote kernel matrix corresponding to the dataset and $\mathbf{A} \in \mathsf{R}^{\mathcal{Q}\times r}$ is the hash projection matrix and $\mathcal{Q}$ is the number of anchors (i.e., a subset of dataset), see more details in [7].

In (1), constraints $\mathbf{B}\mathbf{B}^\top = \mathbf{I}_p$ and $\mathbf{B}\mathbf{1}_n = \mathbf{0}_p$ are uncorrelated and balance conditions, respectively, which are used to encourage compact and balanced binary codes. Particularly, balance condition indicates that each bit has about one-half chance of being 1 or -1 and uncorrelated condition indicates that hash codes to different samples are uncorrelated [8].

Although such three strong constraints imposed on (1) are known to be beneficial for hash coding, these constraints make optimization problem (1) to be intractable and even be 'NP-hard' [6]. In order to make this optimization problem easier to solve in practice, most of the existing algorithms try to address a relatively easier version by adopting the following relaxation strategies: (1), discarding the bit uncorrelated or the bit balance constraints directly; (2), penalizing the uncorrelated (orthogonality) and balance constraints into the objective function as a regularization term; (3), approximating the non-convex optimization problem using a convex one; (4), relaxing the discrete binary conditions to a continuous constraints, then round the continuous solutions to obtain the binary coding.

As mentioned in [8, 9], such compromise strategies might result in large hash coding errors accumulation, thus less accurate. As far as we know, there are only a few methods/models have been proposed to solve hash coding optimization problem (1) that consider all three strong constraints simultaneously. These methods/models include spectral hashing (SH) model [6], discrete graph hashing (DGH) model [9], discrete proximal linearized minimization (DPLM) method [10] and strongly constrained discrete hashing (SCDH) model/methods [7].

In this paper, we consider all the constraints in (1) and propose a penalty function semi-continuous thresholding (PFSCT) hash coding method to solve a generalized hash coding model, where a penalty function method is used to solve the orthogonality constraints optimization subproblem without the eigen-decompostion to large-scale matrix. To reduce quantization error caused by continuous relaxation to discrete binary constraints, a semi-continuous hard thresholding method is established to solve the BDB constraints $\Omega = \{\mathbf{B} \mid \mathbf{B}\mathbf{1}_n = \mathbf{0}_p, \mathbf{B} \in \{-1, 1\}^{p\times n}\}$ subproblem. In addition, we theoretically analyze the equivalence between relaxed problem and original problem for some special objective function. The computational complexity to this alternative iterative optimization method is low and extensive numerical experiments are conducted to evaluate effectiveness of the proposed algorithm.

The remainder of this paper is organized as follows. Section 2 discusses related work regarding tackling different hashing constraints. Section 3 describes the proposed penalty function semi-continuous thresholding methods and the solving algorithm. Experiment results and analysis are presented in Section 4, followed by the conclusions and future work in Section 5.

2. Related works

As discussed above, discrete binary constraint $\mathbf{B} \in \{-1, 1\}^{p \times n}$ is necessary to hash coding problem, but will make the problem to be NP-hard. The methods used in [6, 11] completely discard the discrete binary constraint at first to solve a continuous trackable optimization problem with the remaining conditions, and using thresholding method to binarize the continuous features to obtain the hashing code. In deep learning based hash coding methods [12, 13], the *sgn* or adaptive Tanh functions are used to handle binary constraint. The methods proposed in [14, 15, 16] are to handle the discrete optimization subproblems using a explicitly closed-form formula. However, all of these methods still have the quantization error accumulation problem, especially for long-length codes case [8, 17] due to large span between discrete binary variable and relaxed continuous variable. Very recently, without discarding but relaxing discrete $\mathbf{B} \in \{-1, 1\}^{p \times n}$ into a continuous box constraint $[-1, 1]^{p \times n}$, the methods in [18, 19, 20] make the integer (discrete) optimization problem to be a smooth optimization problem that can be easily solved.

The balanced constraints $\mathbf{B}\mathbf{1}_n = \mathbf{0}_p$, in the discrete case, is used to guarantee the binary hash code are uniformly distributed; in the continuous case, however, it is used to avoid the trivial features solution when performing low-dimension embedding using eigen-decompostion [21]. On the other hand, when balanced constraints are satisfied, the information entropy of hash code can reach the maximum. Generally, there exists three techniques to deal with balanced constraints for binary hash coding problem (1). The first one is to penalize $\mathbf{B}\mathbf{1}_n = \mathbf{0}_p$ constraints directly into the objective function [10] using the Lagrangian multiplier or penalize them into the objective function by maximizing the variance for the k^{th} bit [14, 22]; the second method is the continuous relaxation to balanced constraints and to impose it on the eigenvalue decomposition to avoid the trivial solution [6, 11]; at last, the hash coding models in [23, 15, 17] do not consider the balanced constraints at all, which they believe that these constraints are too strict to hash coding, thus discard them for algorithm feasibility. Inspired by the recently developed discrete hashing learning algorithms [16, 24] that based on the methods solving Binary Quadratic Programming (BQP) [25], we propose a semi-continuous approach to relax the discrete binary constraint and the balanced constraints together, see more details in section 3.

On the other hand, the discrete uncorrelated constraints $\mathbf{B}\mathbf{B}^\top = \mathbf{I}_p$ in (1) can be relaxed as the so-called continuous orthogonality constraints, i.e., $\mathbf{B} \in \mathcal{S}_{p,n} := \{\mathbf{C} \in \mathsf{R}^{p\times n} | \mathbf{C}\mathbf{C}^\top = \mathbf{I}_p\}$ which is usually referred to as the compact $p \times n$ Stiefel manifold. The most common method to handle the orthogonality constraints to problem (1) is to penalize it directly into the objective function [10, 14, 19]. However, this is an inexact penalty that depends on the choice of the penalty parameter, thus is difficult to guarantee the accuracy of the solution. Other ways to deal with orthogonality constraints include spectral decomposition [11, 26], or performing projection directly onto $\mathcal{S}_{p,n}$ [23], but with high computational complexity. In the field of optimization theory, various optimization algorithms have been proposed to solve the problems restricted only to orthogonality constraints, such as the penalty methods [27, 28, 29], gradient-based method [30], splitting method [31], second-order methods [32, 33]. More details can be found in a recent survey literature [34]. Very recently, based on the ideas of augmented Lagrangian method, a first-order infeasible approach named proximal linearized augmented Lagrangian algorithm (PLAM) and its column-wise normalization version (PCAL) for optimization problems with orthogonality constraints is proposed in [28, 35, 36] by using a closed-form expression to update the augmented Lagrangian multiplier. A variant to these algorithms to tackle the uncorrelated constraint in hash coding optimization (1) is proposed in this article, and experiments also have shown its effectiveness, see more details in section 3.

3. Penalty function semi-continuous thresholding (PFSCT) methods

In this section, we first reformulate the hash coding constrained optimization problem (1) into a modified penalty model subject to a compact convex constraint. After that, we introduce a continuous auxiliary variable to bridge this continuous compact convex constraints and the BDB constraint set Ω, where an equivalence objective function is established in the sense that they share similar stationary points. To further reduce the quantization error caused by the BDB constraint Ω, we relax it using a semi-continuous method, and also establish the equivalence between the original optimization (1) and the relaxed optimization problem. In the following, we suppose that the objective function $\mathcal{F}(\mathbf{B})$ satisfying the the following assumption

Assumption 1. *$\mathcal{F}$ is differentiable and $\nabla\mathcal{F}$ is locally Lipschitz continuous.*

Most of objective functions of hash coding in literature satisfy this assumption because Euclidean distance often being used to measure the error, thus is quadratic.

3.1. Model analysis

In general, hash coding problems can be viewed as a two-step process model: the first step is to extract low-dimension continuous feature from the data set; the second step is to convert the continuous feature into discrete hash codes. In other words, hash coding problems are equivalent to a two-step projection processes, one is to project dataset into a low-dimensional feature space, and the other process is to project feature into Hamming space. Motivated by this observation, we reformulate the hash coding problem (1) into the following optimization by introducing an continuous auxiliary variable

$$\min_{\mathbf{C}\in\mathcal{S}_{p,n},\mathbf{B}\in\Omega} \widetilde{\mathcal{F}}(\mathbf{C},\mathbf{B}) = \mathcal{F}(\mathbf{C},\mathbf{B}) + \frac{\mu}{2}\|\mathbf{C}-\mathbf{B}\|_F^2 \quad , \tag{4}$$

where $\mathbf{C}$ is the continuous auxiliary variable, $\mu \to \infty$ is the penalty parameter, $\mathcal{S}_{p,n} := \{\mathbf{C} \in \mathsf{R}^{p\times n} | \mathbf{C}\mathbf{C}^\top = \mathbf{I}_p\}$ is compact $p \times n$ Stiefel manifold and $\Omega := \{\mathbf{B} \mid \mathbf{B}\mathbf{1}_n = \mathbf{0}_p, \mathbf{B} \in \{-1, 1\}^{p\times n}\}$ is the BDB constraints set. The term $\|\mathbf{C} - \mathbf{B}\|_F^2$ is used to measure the quantization error between the continuous feature $\mathbf{C}$ and the discrete binary hashing code $\mathbf{B}$. Generally, alternative iterative optimization algorithm can be exploited to solve this optimization problem (4):

$$\text{Step 1:} \quad \mathbf{C}^{k+1} = \arg\min_{\mathbf{C}\in\mathcal{S}_{p,n}} \widetilde{\mathcal{F}}(\mathbf{C}, \mathbf{B}^k), \tag{5}$$

$$\text{Step 2:} \quad \mathbf{B}^{k+1} = \arg\min_{\mathbf{B}\in\Omega} \widetilde{\mathcal{F}}(\mathbf{C}^{k+1}, \mathbf{B}). \tag{6}$$

However, as above-mentioned, the orthogonality constraint optimization subproblem (5) is nonconvex and maybe are NP-hard in some particular cases, and the BDB constraint optimization subproblem (6) will result in an integer programing, which is also nonconvex and NP-hard. In the following, we will establish algorithms to efficiently tackle these two subproblems, respectively.

3.2. Penalty function method to orthogonality constraint optimization subproblem (5)

Suppose that $\mathbf{B}$ is fixed in orthogonality constraint optimization subproblem (5), and for convenience, we rewrite it as follows

$$\min_{\mathbf{C}} \widetilde{\mathcal{F}}(\mathbf{C}, \mathbf{B}) := \mathcal{F}(\mathbf{C}, \mathbf{B}) + \frac{\mu}{2}\|\mathbf{C} - \mathbf{B}\|_F^2 \quad \text{s.t. } \mathbf{C}\mathbf{C}^\top = \mathbf{I}_p. \tag{7}$$

Inspired by the conclusion proved by Wen [29] and the PCAL method [36, 35], we propose to solve the following optimization problem that use penalty function formulation of (7) to be objective function and imposed with convex constraint $\mathcal{M}$ defined in [36],

$$\min_{\mathbf{C}\in\mathcal{M}} \mathcal{L}(\mathbf{C}) := \widetilde{\mathcal{F}}(\mathbf{C}, \mathbf{B}) + \frac{\beta}{4}||\mathbf{C}\mathbf{C}^\top - \mathbf{I}_p||_F^2, \tag{8}$$

where

$$\mathcal{M} := \{\mathbf{C} \in \mathsf{R}^{p\times n} | ||\mathbf{C}||_F \leq K\}. \tag{9}$$

In addition, $K > \sqrt{p}$ and p is the dimension of feature, and for any $p \times n$ matrix $\widetilde{\mathbf{C}}$, a simple column-wise standardization like $\frac{K}{||\widetilde{\mathbf{C}}||_F}\widetilde{\mathbf{C}}$ can make this new matrix to satisfy constraint $\mathcal{M}$.

The equivalence between the orthogonal constraint optimization subproblem (5) and the corresponding penalty function optimization problem (8) can be established based on **Theorem 2.1** in [29] and **Theorem 3.3** in [36], see more details in these literature.

3.3. Semi-continuous thresholding method to BDB constraint optimization subproblem (6)

In this section, we will establish a method to solve BDB constraint optimization subproblem (6) efficiently and accurately. Given $\overline{\mathbf{C}}$ that comes from (5), the binary constraint optimization subproblem (6) may be solved by minimizing the following objective function involving the quantization error between continuous variable $\overline{\mathbf{C}}$ and binary discrete variable $\mathbf{B}$,

$$\mathbf{B}^* = \arg\min_{\mathbf{B}\in\Omega} \mathcal{F}(\overline{\mathbf{C}}, \mathbf{B}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \mathbf{B}\|_F^2, \tag{10}$$

where Ω is the BDB constraint. Generally, there exist two difficulties when we try to solve (10). Firstly, how to reduce the large quantization errors efficiently between the continuous variable $\overline{\mathbf{C}}$ and the quantified binary discrete variable $\mathbf{B}$? Secondly, how to deal with the binary constraint and the balanced constraints? In order to overcome these two problems, we will reformulate (10) into an extended semi-continuous optimization problem based on the following semi-continuous definition.

Definition 1. *We call scalar variable x to be semi-continuous, if it satisfies $x \in \{0\}\cup[\delta, +\infty)$ with $\delta > 0$. If each element x_i of a vector $\mathbf{x}$ is semi-continuous, we call it a semi-continuous vector. Matrix $\mathbf{B}$ is semi-continuous if its each element is semi-continuous variable.*

Compared with the quantization errors that measures the continuous $\overline{\mathbf{C}}$ and discrete variable $\mathbf{B}$, we hope that a relaxed semi-continuous variable to discrete variable $\mathbf{B}$ can be more accurate to approximate the continuous $\overline{\mathbf{C}}$, thereby reducing the quantization errors. However, when introducing the semi-continuous variable to problem (10), how to maintain the constraints, for example the balanced constraints, to be true is challengeable.

One interesting fact is the output to iterative hard thresholding (IHT) function enjoy the semi-continuous property. The hard thresholding function that performing projection onto the ℓ_0 sphere aims at solving the non-convex and non-smooth optimization problem as follow,

$$\arg\min_{\mathbf{x}} f(\mathbf{x}),\ s.t.\ ||\mathbf{x}||_0 = k$$

where $f(\mathbf{x})$ is gradient Lipschitz continuous and $\mathbf{x}$ is n-dimensional vectors, $||\mathbf{x}||_0$ is the ℓ_0-norm to measure the number of the nonzero entry. Solution to this problem can iterate the hard thresholding function defined by

$$x^{k+1} = \text{hard}(y, \lambda) = \begin{cases} y, |y| \geq \lambda \\ 0, |y| < \lambda \end{cases} \tag{11}$$

where y is every element of vector $\mathbf{y} = \mathbf{x}^k - \alpha \nabla_{\mathbf{x}} f(\mathbf{x}^k)$ and λ is regularization parameter that is related to k, and is generally set λ to the k^{th} largest absolute value of $\mathbf{y}$. It can be find that $\mathbf{x}^* \in \{(-\infty, -\lambda] \cup \{0\} \cup [\lambda, +\infty)\}^n$ admits certain property of semi-continuous.

On the other hand, notice that the balanced constraint $\mathbf{B}\mathbf{1}_n = 0$ in problem (10) implies the number of 1 and -1 in one bit should be uniform, and also inspired by the fact that the ℓ_0 norm represents the number of non-zero elements in a vector, we propose to apply the ℓ_0 norm to express balanced constraint. In fact, for $\mathbf{B} \in \{-1, 1\}^{p \times n}$, the balanced constraint can be rewritten by

$$||\mathbf{b}_i^\top + \mathbf{1}_n||_0 = \frac{n}{2}, \tag{12}$$

where $\mathbf{b}_i$ is the i-th row of binary discrete matrix $\mathbf{B}$. Then, the problem (10) can be reformulated as follow

$$\begin{aligned} \mathbf{B}^* = \arg\min_{\mathbf{B}} \mathcal{F}(\overline{\mathbf{C}}, \mathbf{B}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \mathbf{B}\|_F^2 \\ \text{s.t. } ||\mathbf{b}_i^\top + \mathbf{1}_n||_0 = \frac{n}{2}, i = 1, ...p, \mathbf{B} \in \{-1, 1\}^{p \times n} \end{aligned}. \tag{13}$$

Although equation (12) can be seen as a relaxation of balanced constraint, the existence of $\mathbf{B} \in \{-1, 1\}^{p\times n}$ makes this relaxation tight to the original constraint, thus this problem is strictly equivalent to (10). One possible algorithm to solve (13) is first to solve a continuous relaxation optimization problem using hard thresholding method directly, i.e.,

$$\mathbf{B}^* = sgn(\text{hard}(\overline{\mathbf{B}}, \lambda)). \tag{14}$$

where $\overline{\mathbf{B}} = \widetilde{\mathbf{B}}^k - \alpha\nabla_{\widetilde{\mathbf{B}}}\mathcal{G}(\widetilde{\mathbf{B}}^k)$.

The reasonable side of this algorithm comes from an interesting observation that the hard threshold function and the *sgn* function that commonly used in hash coding share certain common property. In fact, the hard thresholding function converts continuous variable into semi-continuous variable, while *sgn* function converts continuous variable into discrete variable. Based on these observations, we claim that the combination operator *sgn*(hard(·)) may efficiently reduce the quantization error by avoiding the directly discretization to the continuous variables.

To be more consistent, we propose a semi-hard thresholding function and a semi-*sgn* function to replace the hard thresholding function and *sgn* function in (14), respectively. Reformulate (13) as the following equivalent optimization problem by replacing $\mathbf{B}$ with $\mathbf{Y} \in \{0, 1\}^{p\times n}$, which each element -1 in $\mathbf{B}$ is replaced by 0.

$$\begin{aligned} &\mathbf{Y}^* = \arg\min_{\mathbf{Y}} \mathcal{F}(\overline{\mathbf{C}}, \mathbf{Y}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \mathbf{Y}\|_F^2 \\ &\text{s.t. } \|\mathbf{y}_i\|_0 = \frac{n}{2}, i = 1, ...p, \mathbf{Y} \in \{0, 1\}^{p\times n} \end{aligned}, \tag{15}$$

where $\mathbf{y}_i$ is the ith row of matrix $\mathbf{Y}$. In this case, we can view this optimization model as a *rigid* two-classification problem to the entries of $\mathbf{Y}$, which the binary constraint $\mathbf{Y} \in \{0, 1\}^{p\times n}$ indicates the entries of $\mathbf{Y}$ belongs to class either 1 or 0, and the constraint $\|\mathbf{y}_i\|_0 = \frac{n}{2}, i = 1, ...n$ requires this classification is *rigid*, i.e. the number of two classes must be equal as shown in Fig.1.(b). However, the solution obtained by (14) does not necessarily meet this requirement. For example, suppose the $\frac{n}{2}$th largest absolute value of a row in $\mathbf{Y}^*$, which is the

Fig. 1: (a) is the possible situation that can be solved by (14) (the number inside and outside the class is not equal), and (b) is the ideal situation under the action of two constraints (the number inside and outside the class is equal).

result of the gradient descent, is $\gamma > 0$, there may be k negative numbers ($k \leq \frac{n}{2}$) whose absolute value is greater than γ. Then by (14) and due to the *sgn* function, the number of -1 in one row of $\mathbf{Y}^*$ will be ($n - \frac{n}{2} + k \leq \frac{n}{2}$), as shown in Fig.1.(a), where taking $n = 12$ and $k = 2$ as an example. For example, when a row in $\mathbf{Y}^*$ is $[-7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4]$, by (14) it will be $[-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1]$. Therefore, (14) will destroy the balanced constraint. And, it can also be seen that there is no difference between -0.1 and -10 for *sgn*. Of course, we did not consider the more special case where 0 exceeds $\frac{n}{2}$ in a certain row of $\mathbf{Y}^*$, because in this case we cannot obtain a solution that satisfies the balanced constraint by solving a specific operator.

The semi-hard thresholding function is defined by

$$\text{hard}_{semi}(y, \delta) = \begin{cases} y, & y \geq \delta \\ 0, & y < \delta \end{cases}, \tag{16}$$

where y is the element of a row vector $\mathbf{y}$ from $\mathbf{Y}$ and δ is set to the k^{th} largest value of vector $\mathbf{y}$. See Fig.2 for setting $\delta = -1$. The semi-*sgn* function is defined by

$$sgn_{semi}(x) = \begin{cases} 1, & x \neq 0 \\ -1, & x = 0 \end{cases}. \tag{17}$$

In this way, we can relax the constraints $\mathbf{Y} \in \{0,1\}^{p \times n}$ through the semi-

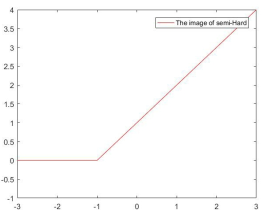


Fig. 2: The image of the semi-hard, when $\delta = -1$.

continuous feature, asthe following semi-continuous optimization problem

$$\begin{aligned} \mathbf{Y}^{k+1} =& \arg\min_{\mathbf{Y}} \ \mathcal{G}(\mathbf{Y}) := \mathcal{F}(\overline{\mathbf{C}}, \mathbf{Y}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \mathbf{Y}\|_F^2 \\ & \text{s.t. } \|\mathbf{y}_i\|_0 = \frac{n}{2}, i = 1, ...p, \mathbf{Y} \in \Omega_\delta(0) \end{aligned}, \tag{18}$$

where $\mathbf{y}_i$ is the i-th row of $\mathbf{Y}$ and $\Omega_\delta(0)$ is an extended semi-continuous domain defined as follows

$$\Omega_\delta(0) \in \begin{cases} \{\{0\} \cup [\delta, +\infty)\}^{p\times n} & \text{if } \delta \geq 0 \\ \{[\delta, 0) \cup \{0\} \cup (0, +\infty)\}^{p\times n} & \text{if } \delta < 0 \end{cases}, \tag{19}$$

and δ is the $\frac{n}{2}$th largest value of a row in $\overline{\mathbf{Y}}$ that is the gradient descent version of $\mathbf{Y}$ as follows $\overline{\mathbf{Y}} = \mathbf{Y}^k - \alpha\nabla_{\mathbf{Y}}\mathcal{G}(\mathbf{Y}^k)$. Note that the reason why it is not written as a continuous interval here is because $\{0\}$ does not mean that the original value is 0 when $\delta < 0$. In this case, by the forward-backward optimization algorithm [37], solution to (18) in one iteration can be expressed by

$$\mathbf{Y}^{k+1} = \text{hard}_{semi}(\overline{\mathbf{Y}}, \delta). \tag{20}$$

And the final hash code can be achieved by applying semi-*sgn* (17) to solution $\mathbf{Y}^*$ of (18), i.e., $\mathbf{B} = sgn_{semi}(\mathbf{Y}^*)$. The overall algorithm is summarized in Algorithm 1.

Let's look at the example we gave above. The solution obtained by (20) is $[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]$, which must satisfy the balance constraint. At last, we need to emphasize that semi-continuous optimization problem (18) can not only be solved efficiently using semi-had thresholding function (20) but also can be solved with less quantization error since the extended semi-continuous relaxation (19) to discrete binary constraints.

3.4. Equivalence analysis

To analyze whether the problem (1) can be solved in the form of the problem (18), we only need to verify whether the solution obtained by (20) is consistent with the solution of the problem (1). We first give a definition of extended support.

Definition 2. *For a vector* $\mathbf{y}$*, supp*$(\mathbf{y})$ *is the position of* 1 *element in* $\mathbf{y}$ *that applied the sgn function.*

Lemma 1. *Suppose that* $\Omega_1 = supp(\mathbf{x}) = \{\omega_1, \omega_2, ..., \omega_k\}$ *is the support of the original variable and* $\Omega_2 = supp(\mathbf{y}) = \{\gamma_1, \gamma_2, ..., \gamma_k\}$*, where* $\mathbf{y} = sgn_{semi}(\mathbf{x})$*. Then, we have* $\Omega_1 = \Omega_2$*.*

Proof 1. *According to Definition 2 of supp and* (17) sgn_{semi}*, this lemma is very easy to obtain.*

Lemma 1 shows that the sgn_{semi} function (17) will not change the position of the support of the variable, which means that the sgn_{semi} function owns the same discreteness capability of the negative semi-axis as the sign function.

Theorem 1. *Solution of* (18) *obtained by* (20) *is equivalent to that of the original problem* (1)*.*

Proof 2. *To prove this theorem, we need to prove that the solution of* (18) *obtained by* (20) *is the solution of the original problem* (1)*. For convenience, let* $\mathbf{B}^*$ *be the close-formed solution [7] of the original problem and* $\mathbf{B}^{'}$ *be the solution obtained by* (20)*. According to the results in paper [29], when* μ *is chosen properly, it is easy to obtain that* (1) *is equivalent to* (4)*. With the help of* ***Theorem 4.2*** *in [35], we can prove the equivalence for any fixed* $\mathbf{B}$*, which implies that solution to* (10) *is equivalent to solution to* (1)*. On the other hand, even though* (12) *can be seen as a relaxation to balanced constraint, the existence of* $\mathbf{B} \in \{-1, 1\}^{p\times n}$ *makes this relaxation tight to the original constraint, thus this problem is strictly equivalent to* (10)*. Therefore supp*$(\mathbf{Y}) =$ *supp*$(\mathbf{B})$ *where* $\mathbf{Y} \in \{0, 1\}^{p\times n}$ *and* $\mathbf{B}$ *is* $\mathbf{B}^*$ *before sgn.*

So, we just need to prove whether the solution to the relaxation problem (18) *of* (10) *is equivalent to the solution of the original problem, i.e. whether* $sgn_{semi}(\mathbf{Y} \in \Omega_\delta(0))$ *is equivalent to* $\mathbf{B}^*$. *Indeed, semi-continuous domain* (19) *only relaxes the value of the non-zero elements of* $\mathbf{Y} \in \{0,1\}^{p\times n}$. *Therefore,* $supp(Y \in \Omega_\delta(0)) = supp(Y \in \{0,1\}^{p\times n})$*, which indicates the conclusion.*

Theorem 1 states that all the equivalent transformations and relaxation to discrete variables that we performed previously do not change the support of the solution. In other words, the semi-continuous variables can not only reduce the "gap" between discrete and continuous variables, but also obtain results equivalent to the original problem.

Algorithm 1 The PFSCT algorithm

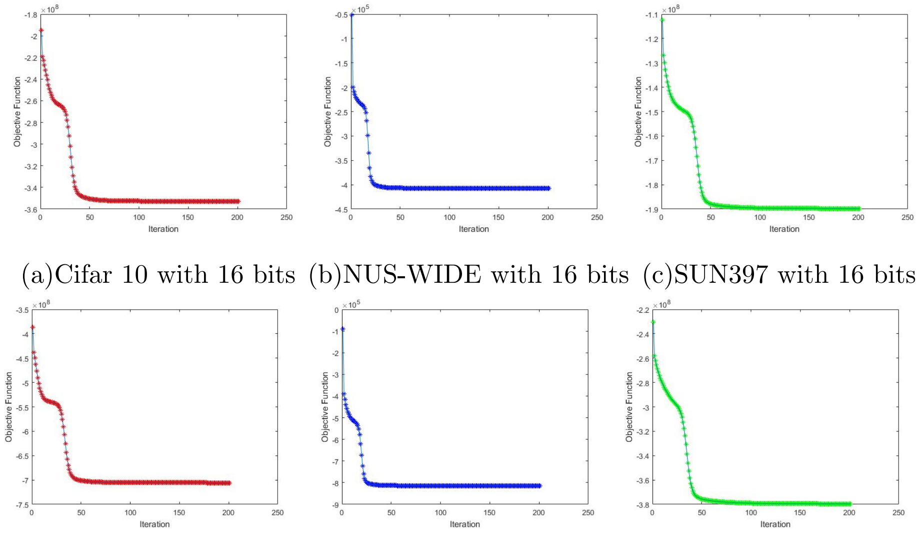
```
Require: Randomly choose $C_0$ on Stiefel manifold and $B_0$ satisfying the constraint, set $K > \sqrt{p}$, $\sigma > 1$, $\beta$ is a constant and $k = 0$
  while not convergence do
    (C-Step): Compute $\widetilde{\mathbf{C}}^{k+1} = \mathbf{C}^k - \gamma_k \nabla_{\mathbf{C}} \mathcal{L}(\mathbf{C}^k, \mathbf{B}^k)$,
    if $||\widetilde{C}^{k+1}||_F > K$ then
      Compute $\mathbf{C}^{k+1} = \frac{K}{||\widetilde{\mathbf{C}}^{k+1}||_2} \widetilde{\mathbf{C}}^{k+1}$.
    else
      $C^{k+1} = \widetilde{C}^{k+1}$
    end if
    (B-Step): Update $\mathbf{B}^{k+1} = \mathrm{hard}_{semi}(\overline{\mathbf{B}}, \delta)$,
    ($\mu$-Step): $\mu^{k+1} = \sigma \mu^k$
  end while
  Get the hash code by $Hash = sgn_{semi}(\mathbf{B}^*)$,
```

3.5. Convergence of the proposed algorithm

From **Algorithm 1**, we can prove that the value of objective function $\mathcal{L}(\mathbf{C}^k, \mathbf{B}^k)$ monotonously non-increasing for each iterative points, i.e., satisfies

$$\mathcal{L}(\mathbf{C}^k, \mathbf{B}^k) \geq \mathcal{L}(\mathbf{C}^{k+1}, \mathbf{B}^k) \geq \mathcal{L}(\mathbf{C}^{k+1}, \mathbf{B}^{k+1}).$$

Under some appropriate assumptions to objective function, for example, is low-bounded and proper, we may prove the proposed alternative iterative algorithm is convergence theoretically. And, in the following section, the extensive experiments also numerically show that our proposed algorithm is convergence.



(a)Cifar 10 with 16 bits (b)NUS-WIDE with 16 bits (c)SUN397 with 16 bits

(d)Cifar10 with 32 bits (e)NUS-WIDE with 32 bits (f)SUN397 with 32 bits

Fig. 3: The objective function value of unsupervised hash model solved by **Algorithm 2**.

4. Numerical Experiment

In this section, we conduct experiments using the proposed PFSCT based on the unsupervised spectral graph hash model [6] and the supervised strongly constrained discrete hashing model [7], respectively. All the experiments are conducted on 3 widely-used datasets: Cifer10, NUS-WIDE and SUN397. We conduct experiments on these two hashing models just because both of these models simultaneously consider all the constraint conditions (i.e., discrete binary constraint, balanced constraint and discrete uncorrelated constraint). We evaluate the proposed PFSCT approach by comparing it with algorithms SH [6] and DPLM [10] for solving the unsupervised model [6], also comparing it with the kernelized algorithm of SCDH(SCDH$_k$) [7] and DPLM [10] for solving the supervised strongly constrained discrete hashing model [7]. To further analyze the effectiveness of the proposed approach, we also compare PFSCT algorithm, which solving the supervised model, with several baseline methods, including unsupervised methods LSH [1], PCAH [22], DSH [38] and supervised methods KSH [39], SDH [17], FastH [40]. Finally, the experiments that using the 512-dimensional hand-crafted image feature of each image in Cifer10 as input to train the hash function are conducted to demonstrate the impact of the image features dataset. In addition, convergence of the algorithms is illustrated by the value of the objective function decreases with the number of iterations. All of the experiments are conducted on the PC with Intel(R) Xeno(R) Gold 5118 CPU @ 2.30GHz and 128GB RAM.

4.1. Datasets

Cifar 10 is labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, etc [6], which consists of 10 different categories of (of size $32 \times 32 \times 3$) images, where each class has 6,000 images. All the classes are completely mutually exclusive, i.e., any picture belongs to one of these categories and is obviously different from other categories. For fair comparison, 50,000 images are randomly selected as the training set(500 images per class), and the remaining 10,000 images as the test set(1000 images per class). In addition, we also use Cifar 10 dataset with the 512-dimensional hand-crafted features to conduct experiments.

NUS-WIDE includes 269,648 images and the associated tags from Flickr, with a total number of 5,018 unique tags [37]. Six types of low-level features extracted from images are recording. For convenience, we first processed the tags of the dataset using the term frequency-inverse document frequency (TF-IDF) cosine similarity method [41] based on the bag of words, which commonly used in natural language processing, to classify the dataset into 10 categories. Also, for fair comparison, we randomly select 30,000 images as training data (300 images per class), and 5,000 images as testing data (50 images per class).

SUN397 contains 108,753 images of 397 categories and is well-sampled categories subsets of **SUN**. As a subset of **SUN**, the number of images of **SUN397** varies across categories and there are at least 100 images per category. We randomly select 32,117 images from 100 categories (also randomly selected from 397 categories) of SUN397 (almost 321 images per class), of which 27,000 images as training data and 5,117 as the testing data.

4.2. Evaluation

As the most frequently used in hash experiments, mean Average Precision (mAP) is used to evaluate the overall performance of the tested algorithms. To compute mAP, we first evaluate the Average Precision(AP), which is a way to summarize the precision-recall curve into a single value representing the average of all precisions. In particular, we compute AP using the following equation

$$\mathrm{AP} = \frac{1}{N}\sum_{k=1}^{K} P(k)\delta(k),$$

where K is the size of dataset, N is the number of relevant data in dataset, $P(k)$ denotes the precision of the top k retrieved data, and $\delta(k) = 1$ if the kth retrieved data is relevant (here, relevant means belonging to the class of the query) and $\delta(k) = 0$. By the definition, mAP is computed by averaging the AP values over all queries in query set, thus is a score between 0 and 1, and the higher the mAP score, the more accurate the tested algorithm. In our result of experiment, the best results are in bold in table.

4.3. Parameter Setting

The program codes of all being compared methods in this paper are provided by the associated article authors. For fair comparison, the input data to all methods are consistent; and the specific parameter settings and precautions of related methods are set according to the recommendations of the associated article authors.

For the proposed PFSCT method in this paper, in order to achieve better convergence to the objective function, we set the maximum iterative number of algorithm to be 50. The parameters in our experiment, we partly refer to the suggestions given in the paper associated to the problem. To obtain better experimental results to the proposed PFSCT, we generally set $\beta = 10$ and $\mu = 1$ of **Algorithm 1** for unsupervised objective function from[6] and $\beta = 1$, $\mu = 5$, $\lambda = 10$ and $\mu = 5$ of **Algorithm 2** for supervised objective function from [7].

4.4. Performance on unsupervised hash model

In this subsection, in order to evaluate the performance of our PFSCT method for solving an unsupervised hash model, we take the unsupervised spectral graph hash model [6] as an example, and compare the performance of PFSCT method with the recently developed DPLM method [10] and the classical SH method [6]. Given n image data $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathsf{R}^{m\times n}$, the spectral graph hash model is given as problem (2), of which $\mathbf{B} \in \{-1, 1\}^{p\times n}$ is the underlying binary hash code matrix, and $\mathbf{L} \in \mathsf{R}^{n\times n}$ is the Laplacian matrix constructed from data $\mathbf{X}$. The Laplacian matrix is defined by $L = D - W$ where W is the pairwise similarity matrix between points and D is diagonal normalization matrix $D_{ii} = \sum_j W_{ij}$. W is calculated using a Gaussian kernel with width σ^2,

$$W_{ij} = \begin{cases} \exp \dfrac{-||x_i - x_j||}{\sigma^2}, x_j \in \mathcal{N}_k(x_i) \\ 0, x_j \notin \mathcal{N}_k(x_i) \end{cases},$$

where $\mathcal{N}_k(x_i)$ is the top k image data most similar to x_i. Then unsupervised hash problem can be rewritten as follow:

$$\min_{\mathbf{C}\in\mathcal{M},\mathbf{B}} \mathcal{L}(\mathbf{C}, \mathbf{B})$$
$$\text{s.t. } \|\mathbf{b}_i + 1\|_0 = \frac{n}{2}, i = 1, ...p, \mathbf{B} + \mathbf{1} \in \Omega_\delta(0),$$

where $\mathcal{L}(\mathbf{C}, \mathbf{B}) = \frac{1}{2}tr(\mathbf{CLC}^\top) + \frac{\beta}{4}||\mathbf{CC}^\top - \mathbf{I}_p||_F^2 + \frac{\mu}{2}||\mathbf{C} - \mathbf{B}||_F^2$ and $\Omega_\delta(0)$ is the extended semi-continuous domain defined by (19), $\mathbf{B}$ is a semi-continuous matrix and $\mathbf{C}$ is its continuous relaxation. By **Algorithm 1**, this problem can be solved efficiently and accurately. In addition, since this unsupervised spectral graph hash objective function is not a joint learning process, i.e., we can not obtain the hash function simultaneously by training this model. To obtain hash code for the data outside the sample, we apply the least square method to train the hash function individually based on the hash code solved above. Particularly, a projection matrix $\mathcal{H} \in \mathsf{R}^{p\times m}$ for $\mathbf{X}$ is obtained by solving the following least square minimizing problem

$$\min_{\mathcal{H}} \|\mathbf{B} - \mathcal{H}\mathbf{X}\|_F^2. \tag{21}$$

In fact, we can also integrate this model into spectral hash model (2) directly to obtain a joint learning model.

We have conducted relevant numerical experiments with 8, 16, 32, 64 bits hash code on above-mentioned Cifar 10, NUS-WIDE and SUN397 datasets, respectively. From evaluation of the objective function value as shown in Fig. 3, it can be seen that the proposed PFSCT algorithm converges very fast, and is very effective for solving unsupervised spectral graph hash model (2) that with all of the three constraints. On the other hand, the mAP values achieved from PFSCT, SH and DPLM algorithms respectively are shown in Table 1, 2 and 3. It can be concluded that the proposed PFSCT algorithm is superior to SH and DPLM algorithms with all code lengths. Particularly, our method achieves high performance with all code lengths for SUN397 and Cifar10 datasets. There is significant gap between the mAP values on SUN397 dataset and that on Cifar10 and NUS-WIDE, part of reason of this phenomena is the resizing and standardization operation to the image in SUN397.

Table 1: The mAP on Cifar10 for unsupervised Hash with different lengths.

Method	8bits	16bits	32bits	64bits
PFSCT	**0.1192**	**0.1211**	**0.1200**	**0.1192**
SH	0.1050	0.1055	0.1087	0.1086
DPLM	0.1020	0.1031	0.1094	0.1101

Table 2: The mAP on NUS-WIDE for unsupervised Hash with different lengths.

Method	8bits	16bits	32bits	64bits
PFSCT	0.1166	**0.1216**	**0.1198**	**0.1192**
SH	**0.1201**	0.1148	0.1140	0.1141
DPLM	0.1128	0.1130	0.1131	0.1132

Table 3: The mAP on SUN397 for unsupervised Hash with different lengths.

Method	8bits	16bits	32bits	64bits
PFSCT	**0.0227**	**0.0223**	**0.0224**	**0.0226**
SH	0.0191	0.0197	0.0200	0.0187
DPLM	0.0194	0.0193	0.0193	0.0192

In addition, one possible flaw of PFSCT is the mAP value is not perfect positive correlation with the hash code length, which also emerges occasionally in SH and DPLM. The reason may be due to the unsupervised nature of spectral graph model itself. In fact, spectral graph hash can be regarded as a binary classification problem based on Laplacian spectral feature, which its effectiveness not only depends on the accuracy of the solving algorithm, but also depends on the constructed Laplacian matrix which describes the geometric structure of the dataset. The common Gaussian kernel based on the Euclidean distance between the samples is often inadequate to hashing tasks. And, the classical spectral learning methods provide parametrization of merely the observable manifold/dataset rather than the desired underlying manifold/dataset [42].

The computational cost of our algorithm to spectral graph hash model (2) mainly comes from the update $\mathbf{B}$ and $\mathbf{C}$ by computing the gradient of the subproblem. For updating $\mathbf{C}$, the computational complexity is $O(tp^2n)$, where p is the length of the hash code and t is the iteration number. For updating $\mathbf{B}$, the computational cost is $O(tpn)$. Such computational complexity is comparable to that of DPLM algorithm, but outperforms the computational complexity of the SH algorithm, which is $O(n^3)$. As stated in [7], if anchors method is used, the computational complexity will be further reduced to $O(mn)$ where m is the number of anchors, which is linear to the data size n.

4.5. Performance for supervised hash model

To evaluate the effectiveness of our algorithm for supervised problem, in this subsection, taking the supervised SCDH model (3) as an example, we compare our algorithm for solving this supervised model with the algorithm DPLM [10] and the SCDH$_k$ [7]. Because the objective function of supervised SCDH model (3) satisfy the Assumption 1, thus, we can also solved it using the paradigm of **Algorithm 2**. Similarity, we reformulate supervised SCDH model using the methods discussed in section 3 as the following optimization problem

$$\begin{gathered}\min_{\mathbf{Z}\in\mathcal{M},\mathbf{B},\mathbf{A}} \mathcal{L}(\mathbf{Z},\mathbf{B},\mathbf{A})\\ \text{s.t. } \|\mathbf{b}_i+1\|_0 = \frac{n}{2}, i=1,...r, \mathbf{B}+\mathbf{1}\in\Omega_\delta(0)\end{gathered}$$

where $\mathcal{L}(\mathbf{Z},\mathbf{B},\mathbf{A}) = \mathcal{F}(\mathbf{Z},\mathbf{B},\mathbf{A}) + \frac{\beta}{4}||\mathbf{Z}^\top\mathbf{Z}-\mathbf{I}_r||_F^2$ and $\mathcal{F}(\mathbf{Z},\mathbf{B},\mathbf{A})$ is the objective function in (3). And kernel matrix $\mathcal{K}_\mathcal{Q}$ is constructed using the original data and its corresponding anchors, see more details in [7]. The matrix variables $\mathbf{B}$ and $\mathbf{Z}$ can still be solved by **Algorithm 1**. Hash projection matrix $\mathbf{A}\in\mathsf{R}^{\mathcal{Q}\times r}$ can be obtained by solving the following optimization problem

$$\min_A ||\mathcal{K}_\mathcal{Q}\mathbf{A}-\mathbf{B}||_F^2 + \frac{\gamma}{\lambda}||\mathbf{A}||_F^2,$$

which is sample and enjoys closed solution

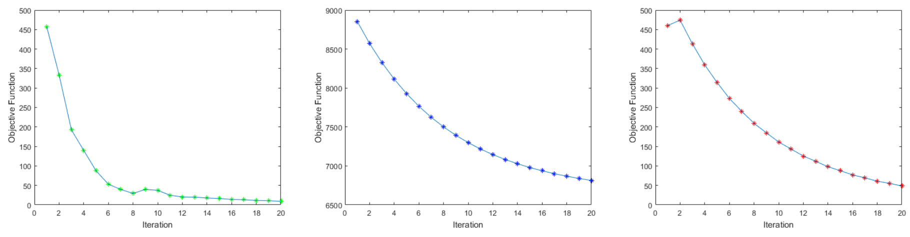
$$\mathbf{A} = (\mathcal{K}_\mathcal{Q}{}^\top\mathcal{K}_\mathcal{Q} + \gamma\mathbf{I_M})^{-1}\mathcal{K}_\mathcal{Q}{}^\top\mathbf{B}. \tag{22}$$

Detailed algorithm shown in **Algorithm 2**.

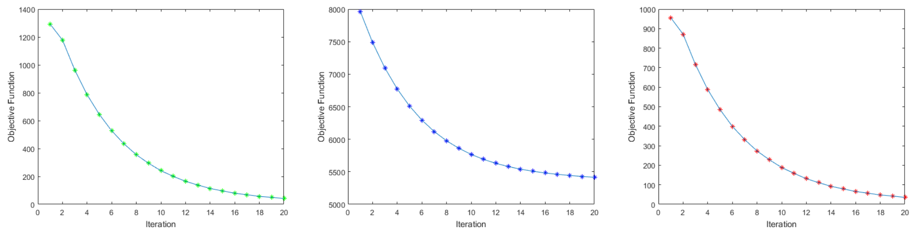
Algorithm 2 The PFSCT algorithm to supervised hashing model

```
Require: Randomly choose $C_0$ on Stiefel manifold and $B_0$ satisfying the con-
    straint, set $K > \sqrt{r}$ and $k = 0$
    while not convergence do
        (A-Step)
        Compute $\mathcal{A}$ by (22)
        Update $\tilde{C}$, $\tilde{B}$ and $\mu$ follow Algorithm 2
    end while
    Get the hash code by $Hash = sgn_{semi}(\mathbf{B}^*)$,
```

Based on the same dataset as being used in unsupervised hash code model, we have conducted supervised hash numerical experiments using **Algorithm 2**. The value of objective function decay very fast seen from Fig.4 for the 8, 16 bits case, which indicates convergence of the algorithm.



(a)Cifar 10 with 8 bits (b)NUS-WIDE with 8 bits (c)SUN397 with 8 bits



(d)Cifar10 with 16 bits (e)NUS-WIDE with 16 bits (f)SUN397 with 16 bits

Fig. 4: The objective function value of supervised hash model solved by **Algorithm 3**.

The mAP values achieved from PFSCT, SCDH$_k$ and DPLM algorithms respectively are shown in Table 4, 5 and 6. Unsurprisingly, compared with the

Table 4: The mAP on Cifar10 for supervised Hash with different lengths.

Method	8bits	16bits	32bits	64bits
SCDH_K	0.6506	0.6509	**0.6526**	**0.6537**
DPLM	0.6508	0.6509	0.6510	0.6513
PFSCT	**0.6508**	**0.6512**	0.6514	0.6516

Table 5: The mAP on NUS-WIDE for supervised Hash with different lengths.

Method	8bits	16bits	32bits	64bits
SCDH_K	0.6680	0.6682	**0.6712**	**0.6727**
DPLM	0.6679	0.6680	0.6682	0.6683
PFSCT	**0.6681**	**0.6690**	0.6689	0.6683

unsupervised hash model, it is shown that the supervised model overwhelmingly outperforms it from the mAP values viewpoint, this is consistent with our intuition and the label information can in general improve the effectiveness of hashing. It can also being seen that the proposed PFSCT is comparable to SCDH_k and is a little superior to DPLM. For the long hash code length, such as 32 bits and 64 bits, SCDH_k methods do better. On the other hand, the total time cost of supervised hash method is significantly less than that of unsupervised hash method.

Table 6: The mAP on SUN397 for supervised Hash with different lengths.

Method	8bits	16bits	32bits	64bits
SCDH_K	0.4851	0.4851	0.4848	**0.4853**
DPLM	0.4850	0.4850	0.4851	0.4852
PFSCT	**0.4851**	**0.4851**	**0.4850**	0.4852

4.6. Performance of feature dataset v.s. original image dataset

To evaluate the impact that applying image features dataset to train hash code instead of original image dataset, in this subsection, we conduct experiment with 8, 16, 32, 64 bits hash code on the 512-dimensional hand-crafted feature dataset of Cifar 10 by using **Algorithm 2** to solve the supervised SCDH model (3). But the kernel $\mathcal{K}_{\mathcal{Q}}$ here is constructed using the 512-dimensional hand-crafted feature dataset and its corresponding anchors, instead of original dataset.

Table 7: The mAP on Cifar10-Gist to supervised model with different lengths.

Method	8bits	16bits	32bits	64bits
SCDH$_K$	0.6984	**0.7012**	**0.7129**	**0.7132**
DPLM	0.6983	0.7004	0.7105	0.7105
PFSCT	**0.6988**	0.7006	0.7105	0.7106

As shown in Table 7, it can be seen that mAP values using feature dataset significantly outperform the mAP values that using original image dataset shown in Table 4. Consider the limited space of paper, we will not illustrate the similar conclusions about the other two datasets. This result also is consistent with our intuition, and actually using semantic feature information of image dataset to describe the similarity between samples in dataset are more accurate than that only using samples itself. We believe that, in the process of establishing hash model, the more structural semantic feature information being used, the more accurate the model is, which is also our future research direction.

4.7. Results of comparison with other baseline methods

In this section, we compare our proposed PFSCT method for solving supervised SCDH model with several baseline methods on Cifar 10 and NUS-WIDE dataset. Among these methods, the LSH [1], PCAH [22], DSH [38] are unsupervised methods, while the KSH [39], SDH [17], FastH [40] are supervised methods. The PCAH and SDH methods have exploited the modeling techniques similar to SH and SCDH, but do not consider balanced constraint and discrete binary constraint, respectively. The LSH, DSH, KSH and FastH methods have established the model using the other techniques, thus only consider the discrete binary constraint.

Table 8: The mAP of baseline Hash methods with different lengths on Cifar10.

Method	8bits	16bits	32bits	64bits
LSH	0.1068	0.1120	0.1085	0.1256
PCAH	0.1061	0.1064	0.1079	0.1069
DSH	0.1155	0.1111	0.1166	0.1280
KSH	0.1669	0.1766	0.1959	0.2081
SDH	0.2074	0.2275	0.2668	0.2905
FastH	0.4362	0.4932	0.5923	0.6433
PFSCT	**0.6508**	**0.6512**	**0.6514**	**0.6516**

Unsurprisingly, the supervised models are superior to the unsupervised method as shown in Table 8 and 9, and our proposed PFSCT algorithm overwhelmingly outperform these methods. The accuracy of hashing process mainly depends on the effectiveness of model and the corresponding solving algorithm. The KSH, FastH methods mainly are inclined to study the application of Kernel technique and the fast algorithm. The LSH and DSH methods apply the probability of collision of simples and entropy theory, respectively, to implement hashing process, which is lack of accuracy. The PCAH and SDH methods only consider part of the constraints in their model, which is also not accurate enough. On the other hand, our proposed PFSCT method effectively reduces the quantization error, and adopts the ideas of *rigid* two-classification to encode the hashing code to dissimilar data, thus can achieve higher mAP values than other methods.

5. Conclusion And Future

In this article, we propose a comprehensive method to solve generalized hash optimization problems imposed with three constraints. A simple and effective inaccurate penalty function algorithm is proposed to deal with the orthogonal constraint optimization subproblem, which may avoiding eigen-decomposition to large-scale matrix. To reduce the quantization error as much as possible, a semi-continuous variable is introduced and a fast iterative semi-hard thresh-

Table 9: The mAP of baseline Hash methods with different lengths on NUS-WIDE.

Method	8bits	16bits	32bits	64bits
LSH	0.1191	0.1215	0.1258	0.1288
PCAH	0.1250	0.1244	0.1217	0.1193
DSH	0.1258	0.1305	0.1277	0.1314
KSH	0.1327	0.1416	0.1463	0.1486
SDH	0.2253	0.2549	0.3044	0.3200
FastH	0.4566	0.5097	0.5474	0.5981
PFSCT	**0.6681**	**0.6690**	**0.6689**	**0.6683**

olding algorithm is established to solve the binary discrete balance constraint subproblems. The extensive experiments illustrate that the proposed algorithms is fast and efficient compared with the state-of-the-art algorithms.

We only numerically show the convergence of the proposed algorithm, thus, rigorous theoretical analysis is our future work. In addition, how to accurately represent the structural semantic features for hash coding problem to large-scale complex dataset that including non-Euclidean or multimodal data will also be considered in our future work.

[1] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, 1999.

[2] Y. Yang, Z. J. Zha, Y. Gao, X. Zhu, T. S. Chua, Exploiting web images for semantic video indexing via robust sample-specific loss, IEEE Transactions on Multimedia 16. `doi:10.1109/TMM.2014.2323014`.

[3] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, J. Yagnik, Fast, accurate detection of 100,000 object classes on a single machine, 2013. `doi:10.1109/CVPR.2013.237`.

[4] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, J. Attenberg, Feature hashing for large scale multitask learning, 2009.

[5] X. Liu, J. He, C. Deng, B. Lang, Collaborative hashing, 2014. doi:10.1109/CVPR.2014.275.

[6] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, 2009.

[7] Y. Chen, Z. Tian, H. Zhang, J. Wang, D. Zhang, Strongly constrained discrete hashing, IEEE Transactions on Image Processing 29. doi:10.1109/TIP.2020.2963952.

[8] J. Wang, T. Zhang, J. Song, N. Sebe, H. T. Shen, A survey on learning to hash, IEEE Transactions on Pattern Analysis and Machine Intelligence 40. doi:10.1109/TPAMI.2017.2699960.

[9] W. Liu, C. Mu, S. Kumar, S. F. Chang, Discrete graph hashing, Vol. 4, 2014.

[10] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, D. Tao, A fast optimization method for general binary code learning, IEEE Transactions on Image Processing 25. doi:10.1109/TIP.2016.2612883.

[11] J. Song, Y. Yang, Z. Huang, H. T. Shen, R. Hong, Multiple feature hashing for real-time large scale near-duplicate video retrieval, 2011. doi:10.1145/2072298.2072354.

[12] J. Lu, V. E. Liong, J. Zhou, Deep hashing for scalable image search, IEEE Transactions on Image Processing 26. doi:10.1109/TIP.2017.2678163.

[13] D. Wang, P. Cui, M. Ou, W. Zhu, Learning compact hash codes for multimodal representations using orthogonal deep structure, IEEE Transactions on Multimedia 17. doi:10.1109/TMM.2015.2455415.

[14] J. Wang, S. Kumar, S. F. Chang, Semi-supervised hashing for scalable image retrieval, 2010. doi:10.1109/CVPR.2010.5539994.

[15] X. Luo, L. Nie, X. He, Y. Wu, Z. D. Chen, X. S. Xu, Fast scalable supervised hashing, 2018. doi:10.1145/3209978.3210035.

[16] X. Luo, P. F. Zhang, Z. Huang, L. Nie, X. S. Xu, Discrete hashing with multiple supervision, IEEE Transactions on Image Processing 28. `doi:10.1109/TIP.2019.2892703`.

[17] F. Shen, C. Shen, W. Liu, H. T. Shen, Supervised discrete hashing, Vol. 07-12-June-2015, 2015. `doi:10.1109/CVPR.2015.7298598`.

[18] B. Wu, B. Ghanem, ℓ_p-box admm: A versatile framework for integer programming, IEEE Transactions on Pattern Analysis and Machine Intelligence 41. `doi:10.1109/TPAMI.2018.2845842`.

[19] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, H. T. Shen, Unsupervised deep hashing with similarity-adaptive and discrete optimization, IEEE Transactions on Pattern Analysis and Machine Intelligence 40. `doi:10.1109/TPAMI.2018.2789887`.

[20] M. A. Carreira-Perpiñán, R. Raziperchikolaei, Hashing with binary autoencoders, Vol. 07-12-June-2015, 2015. `doi:10.1109/CVPR.2015.7298654`.

[21] M. M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, P. Vandergheynst, Geometric deep learning: Going beyond euclidean data (2017). `doi:10.1109/MSP.2017.2693418`.

[22] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 35. `doi:10.1109/TPAMI.2012.193`.

[23] K. Zhao, H. Lu, Y. He, S. Feng, Locality preserving discriminative hashing, 2014. `doi:10.1145/2647868.2654971`.

[24] S. Zhang, J. Li, M. Jiang, P. Yuan, B. Zhang, Scalable discrete supervised multimedia hash learning with clustering, IEEE Transactions on Circuits and Systems for Video Technology 28. `doi:10.1109/TCSVT.2017.2710345`.

[25] R. Yang, New results on some quadratic programming problems (2013).

[26] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, 2010. doi:10.1145/1835449.1835455.

[27] Z. Wen, W. Yin, A feasible method for optimization with orthogonality constraints, Mathematical Programming 142. doi:10.1007/s10107-012-0584-1.

[28] B. Gao, X. Liu, X. Chen, Y. X. Yuan, A new first-order algorithmic framework for optimization problems with orthogonality constraints, SIAM Journal on Optimization 28. doi:10.1137/16M1098759.

[29] Z. Wen, C. Yang, X. Liu, Y. Zhang, Trace-penalty minimization for large-scale eigenspace computation, Journal of Scientific Computing 66. doi:10.1007/s10915-015-0061-0.

[30] J. H. Manton, Optimization algorithms exploiting unitary constraints, IEEE Transactions on Signal Processing 50. doi:10.1109/78.984753.

[31] R. Lai, S. Osher, A splitting method for orthogonality constrained problems, Journal of Scientific Computing 58. doi:10.1007/s10915-013-9740-x.

[32] J. Hu, A. Milzarek, Z. Wen, Y. Yuan, Adaptive quadratically regularized newton method for riemannian optimization, SIAM Journal on Matrix Analysis and Applications 39. doi:10.1137/17M1142478.

[33] A. Edelman, T. A. Arias, S. T. Smith, The geometry of algorithms with orthogonality constraints, SIAM Journal on Matrix Analysis and Applications 20. doi:10.1137/S0895479895290954.

[34] J. Hu, X. Liu, Z. W. Wen, Y. X. Yuan, A brief introduction to manifold optimization, Journal of the Operations Research Society of China 8. doi:10.1007/s40305-020-00295-9.

[35] B. Gao, X. Liu, Y. X. Yuan, Parallelizable algorithms for optimization problems with orthogonality constraints, SIAM Journal on Scientific Computing 41. doi:10.1137/18M1221679.

[36] N. Xiao, X. Liu, Y. xiang Yuan, A class of smooth exact penalty function methods for optimization problems with orthogonality constraints, Optimization Methods and Softwaredoi:10.1080/10556788.2020.1852236.

[37] H. Attouch, J. Bolte, B. F. Svaiter, Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized gauss-seidel methods, Mathematical Programming 137. doi:10.1007/s10107-011-0484-9.

[38] Z. Jin, C. Li, Y. Lin, D. Cai, Density sensitive hashing, IEEE Transactions on Cybernetics 44. doi:10.1109/TCYB.2013.2283497.

[39] W. Liu, J. Wang, R. Ji, Y. G. Jiang, S. F. Chang, Supervised hashing with kernels, 2012. doi:10.1109/CVPR.2012.6247912.

[40] G. Lin, C. Shen, Q. Shi, A. V. D. Hengel, D. Suter, Fast supervised hashing with decision trees for high-dimensional data, 2014. doi:10.1109/CVPR.2014.253.

[41] L. Havrlant, V. Kreinovich, A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation), International Journal of General Systems 46. doi:10.1080/03081079.2017.1291635.

[42] R. Talmon, I. Cohen, S. Gannot, R. R. Coifman, Diffusion maps for signal processing: A deeper look at manifold-learning techniques based on kernels and graphs (2013). doi:10.1109/MSP.2013.2250353.