Penalty function semi-continuous thresholding hashing code learning methods for large-scale data retrieval

Qian Chen[1], Zhengwei Shen[1,2], Zhe Chen[1]
[1]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 to a NP-hard optimization problem. A variety of algorithms have been established to solve it based on multifarious relaxation/discarding techniques to these three constraints, which have either high computational complexity or poor accuracy. Different from exiting algorithms, in this study, we propose a penalty function semi-continuous thresholding(PFSCT) hash coding algorithm that is a fast and accurate framework for solving the hashing model with all these three constraints. Particularly, we firstly divide the whole constraints set into the uncorrelated(orthogonality) constraint and the binary discrete balance constraint two parts. Then, a variant of penalty function model imposed with a relatively regular sphere constraint is established to convex relax the nonconvex uncorrelated(orthogonality) constraint subproblems, and an efficient forward-backward algorithms is used to solve it. Moreover, to solve the binary discrete balance constraint subproblems more accurately, a fast iterative semi-hard thresholding algorithm is proposed based on a semi-continuous relaxation to the binary discrete balance constraint. We also theoretically analyze the equivalence between the relaxed model and the original problems. To evaluate the proposed PFSCT, we performed extensive numerical experiments on three large-scale benchmark datasets, and the mAP results demonstrate comparable performance.

***Index Terms*—Hash.**

I. Introduction

As an efficient dimensionality reducing and information retrieval techniques to high-dimensional data, hash coding has been widely applied to a variety of large-scale information processing and machine learning problems including similarity search[2], [3], image and video retrieval [4], [5], [17], [19], objective detection[6], large scale multitask learning[7], and recommendation system[8], [9]. 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 the 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 usually used to characterize certain weighted average Hamming distance and the associated regularizer. Constraints $\mathbf{B}\mathbf{B}^\top = \mathbf{I}_p$ and $\mathbf{B}\mathbf{1}_n = \mathbf{0}_p$ are binary bit uncorrelated and balance conditions, respectively, which are used to encourage compact and balanced binary code. Bit balance condition indicates that each bit has about one-half chance of being 1 or -1. Bit uncorrelated condition indicates that hash code to different samples are uncorrelated[1].

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[51]. Indeed, discrete binary constraints $\mathbf{B} \in \{-1,1\}^{p\times n}$ imposed on the target hash coding will lead to an integer programing(IP), which is non-convex and NP-complete(thus is NP-hard). Uncorrelated constraints $\mathbf{B}\mathbf{B}^\top = \mathbf{I}_p$(a.k.a. orthogonal constraints in the field of optimization theory) is loaded by non-convex constraints since it is equivalent to Stiefel manifold constraints in real matrix space[10], which will leads to many local minimizers and, in particular, maybe also is NP-hard[14]; meanwhile, orthogonality constraints are numerically expensive to be preserved during iterations algorithm.

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:

- Discarding the bit uncorrelated(orthogonal) or the bit balance constraints directly;
- Penalizing the uncorrelated(orthogonal) and balance constraints into the objective function as a regularization term;
- Approximating the non-convex optimization problem using a convex one;
- Relaxing the discrete binary conditions to a continuous constraints, then round the continuous solutions to obtain the binary coding.

As mentioned in [1], [38], such compromise strategies might result in large hash coding errors accumulation, thus less accurate. However, 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 con-

Manuscript received ******** Corresponding author: Zhengwei Shen

straints simultaneously, which including spectral hashing(SH) model[51], discrete graph hashing(DGH) model [38], discrete proximal linearized minimization(DPLM) method[11] and strongly constrained discrete hashing(SCDH) model[36].

The SH model[51] has proposed to use uncorrelated and balance constraints to learn semantic hashing code for a given dataset. Two steps learning scheme is established, firstly, a continuous Laplacian spectral embedding problem being solved by discarding discrete binary constraints; then sign thresholding the eigenvectors being used to obtain the binary discrete hashing code. However, these techniques are suboptimal especially for a long hash code[1], because of excessively simple sign thresholding operation to the continuous eigenvectors. In addition, the eigen-decomposition to large-scale Laplacian matrix will result in high computational cost and be even worse particularly when adapting iterative algorithm. By designing a real matrix set $\Gamma = \left\{\mathbf{Y} \in \mathsf{R}^{p\times n} \mid \mathbf{Y}^\top \mathbf{1} = \mathbf{0}_p, \mathbf{Y}\mathbf{Y}^\top = \mathbf{I}_p\right\}$, the DGH scheme[38] has established a distance function to measure binary code matrix $\mathbf{B}$ and Γ. This method enforces the discrete constraint directly to achieve binary code matrix $\mathbf{B}$ for avoiding error caused by the continuous relaxation method, but is less accurate because of penalizing the real matrix set Γ into objective function. By penalizing the uncorrelated and balance conditions into the objective function respectively to generate a differential objective function $\mathcal{F}(\mathbf{B})$, the DPLM optimization scheme[11] has proposed to use the iterative forward-backward method to seek a projection operation of the forward result $\mathbf{B}^{(j)} - \frac{1}{\lambda}\nabla\mathcal{F}\left(\mathbf{B}^{(j)}\right)$ onto binary code space $\{-1,1\}^{p\times n}$. Very recently, the SCDH model[36] has established a supervised hash coding model that consider all of the constraints as shown in (1). An real-valued auxiliary variable $\mathbf{Z}$ as an alias of $\mathbf{B}$ is introduced to build a bridge between the discrete and continuous optimization, and an iterative alternative optimization method is established to solve this SCDH model with each subproblem having closed form solution. However, in order to deal with the orthogonal constraint conditions, the singular value decomposition(SVD) to a large scale matrix in each iteration will greatly increase the computational complexity. In addition, penalizing the real-valued auxiliary matrix $\mathbf{Z}$ and the binary code matrix $\mathbf{B}$ into the objective using Euclidean distance $\|\mathbf{Z} - \mathbf{B}\|_F^2$ is inaccurate, and directly minimizing this quantization loss that is a asymmetric distance is impractical from the optimization viewpoints[50].

In this paper, we consider all the constraint conditions in (1) and propose a penalty function semi-continuous thresholding(PFSCT) hash coding method, of which the constraints in (1) are firstly divided into two parts: the uncorrelated constraint part $\mathbf{B}\mathbf{B}^\top = \mathbf{I}_p$ and the binary discrete balance(BDB) constraint parts, i.e. $\Omega = \{\mathbf{B} \mid \mathbf{B}\mathbf{1}_n = \mathbf{0}_p, \mathbf{B} \in \{-1,1\}^{p\times n}\}$. An efficient penalty function method is proposed to solve the uncorrelated (orthogonality) constraints optimization subproblem. By relaxing the BDB constraints into a semi-continuous set, the BDB constraints optimization subproblem is solved by the proposed iterative semi-hard thresholding algorithm. An alternative iterative optimization method is established to efficiently solve the whole optimization model. Such a comprehensive framework will avoid the general projection procedure onto the discrete constraint set when dealing with uncorrelated constraints and give a more precise relaxation of discrete constraints, which can reduce the quantization error during relaxation as much as possible. The main contributions of PFSCT are summarized as follows.

- A smooth exact penalty function method is proposed to solve the orthogonality constraints optimization subproblem. Specifically, a convex compact constraint, which is a convex hull of the nonconvex Stiefel manifold (i.e. the orthogonality constraints), is established to relax original nonconvex constraints; then, forward-backward algorithm is used to solve this problem followed by projecting the iteration sequence on this convex compact constraint. This method is efficient and fast because any gradient method for solving this model is already a second-order method for solving the original optimization; meanwhile, the algorithm does not need to perform eigen-decompostion to large-scale Laplacian matrix which often indicates high computational complexity.
- To reduce quantization error caused by continuous relaxation to discrete binary constraints, a semi-continuous hard thresholding method is established to solve the subproblem that imposed by BDB constraints. An iterative semi-hard thresholding algorithm is established to efficiently solve this semi-continuous subproblem. This method not only may deal with the discrete binary constraints effectively, but also can characterize the distance between the continuous variable $\mathbf{C}$ and the introduced semi-continuous set accurately. In addition, when there is a low error(determined by a given thresholding) between the continuous variable $\mathbf{C}$ and the introduced semi-continuous variable, this semi-discrete variable will be changed into binary discrete codes. This scheme allows the hash code to represent as much information as possible about the low-dimension embedding feature, thus achieve more accurate hash codes.
- We theoretically analyze the equivalence of the relaxed problem and the original problem for some special objective function form. The computational complexity of the proposed two-steps iterative optimization method is low, because all the sub-problems are solved with relatively simple and fast convergence method. The comprehensive numerical experiments and analysis are conducted to evaluate the effectiveness of the proposed algorithm.

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

II. Related works

As discussed in introduction, 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[51], [17], [18], [37], [49], [50] 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[23], [24], [25], the *sgn* or adaptive Tanh functions are used to handle binary constraint. The methods proposed in[19], [27], [28], [36], [47], [48] 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[1], [47] 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 constriant $[-1, 1]^{p\times n}$, the methods in[20], [21], [22] make the discrete hashing problem to be a smooth optimization problem that can be easily solved, but with lower computational efficiency.

The balanced constraints $\mathbf{B1}_n = \mathbf{0}_p$, 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[46]. 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 $\mathbf{B1}_n = \mathbf{0}_p$ constraints directly into the objective function [11] using the Lagrangian multiplier or penalize them into the objective function by maximizing the variance for the k^{th} bit[19], [50]; the second method is the continuous relaxation to balanced constraints and to impose it on the eigenvalue decomposition to avoid the trivial solution[51], [17], [18], [36]; at last, the hash coding models in [12], [27], [47] 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. There is little doubt that balanced constraints are necessary to guarantee the accuracy of the hash coding, but in practice, it is hard to ensure that the features represented by 1-bit hash code are uniformly distributed. In order to overcoming this predicament, and inspired by the recently developed discrete hashing learning algorithms[28], [29] that based on the methods solving Binary Quadratic Programming (BQP)[30], we propose a semi-continuous approach to relax the discrete binary constraint and the balanced constraints together, see more details in section III.

On the other hand, the discrete uncorrelated constraints $\mathbf{BB}^\top = \mathbf{I}_p$ in (1) can be relaxed as the so-called continuous orthogonal constraints, i.e. $\mathbf{B} \in \mathcal{S}_{p,n} := \{\mathbf{C} \in \mathsf{R}^{p\times n} | \mathbf{CC}^\top = \mathbf{I}_p\}$ which is usually referred to as the compact $p \times n$ Stiefel manifold. The problems with such constraints usually are difficult to be solved because the orthogonal constraints are not only non-convex but numerically expensive to preserve during iterations. The most common method to handle the orthogonal constraints to problem (1) is to penalize it directly into the objective function[11], [19], [21], [37], [38]. 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 orthogonal constraints include spectral decomposition[51], [17], [18], or performing projection directly onto $\mathcal{S}_{p,n}$[12], but with high computational complexity. In the field of optimization theory, various optimization algorithms have been proposed to solve the problems restricted only to orthogonal constraints, such as the penalty methods [14], [15], [16], gradient-based method[41], [42], splitting method[43], second-order methods[44], [45]. More details can be found in a recent literature review[13]. 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[15], [31], [32] by using a closed-form expression to update the augmented Lagrangian multiplier. To preserve the orthogonality, the orthonormalization procedure is only invoked once at the last step of PLAM and PCAL algorithm. Using these algorithms to tackle the uncorrelated constraint in hash coding optimization (1) in this paper proved to be very efficient, see more details in section III.

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

In this section, focusing on the uncorrelated(orthogonality) constraints and using an efficient infeasible approaches based on the penalty function methods[31], [32], 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. An equivalence for a special objective function form between this reformulation and the original problem(1) is established in the sense that they share the same second-order stationary points, and hence, the global minimizers under certain mild conditions. 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 satisfying this assumption because Euclidean distance often being used to measure the error, thus is quadratic, in most cases. For example, the unsupervised spectral graph hash model that being given in [51]

$$\begin{aligned} &\min_{\mathbf{B}} \frac{1}{2} tr(\mathbf{BLB}^\top) \\ &s.t. \quad \mathbf{BB}^\top = \mathbf{I}_p, \mathbf{B1}_n = \mathbf{0}_p, \mathbf{B} \in \{-1, 1\}^{p\times n} \end{aligned}, \tag{2}$$

where $\mathbf{L}$ is the Laplacian matrix of data, p is the length of hash code of each data, n is the number of given data; and the supervised SCDH model being established in [36]

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{A},\mathbf{Z}} \mathcal{F}(\mathbf{B},\mathbf{A},\mathbf{Z}) \\ s.t. \quad &\begin{cases} \mathbf{B} \in \{-1,1\}^{n\times r} \\ \mathbf{Z} \in \mathsf{R}^{n\times r}, \mathbf{Z}^\top \mathbf{1}_n = \mathbf{0}_r, \mathbf{Z}^\top \mathbf{Z} = n\cdot \mathbf{I}_r \end{cases}, \end{aligned} \tag{3}$$

where objective function $\mathcal{F}(\mathbf{B},\mathbf{A},\mathbf{Z})$ is

$$||r\cdot \mathbf{S} - \mathbf{B}\mathbf{Z}^\top||_F^2 + \lambda||\mathcal{K}_{\mathcal{Q}}\mathbf{A} - \mathbf{B}||_F^2 + \alpha||\mathbf{B}-\mathbf{Z}||_F^2 + \beta||\mathbf{A}||_F^2$$

and r is the length of hash code, n is the number of given data, $\mathbf{S} \in [-1,1]^{n\times n}$ 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 [36].

A. 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 (supervised or unsupervised) 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

$$\begin{aligned} \min_{\mathbf{C},\mathbf{B}} \widetilde{\mathcal{F}}(\mathbf{C},\mathbf{B}) = \mathcal{F}(\mathbf{C},\mathbf{B}) + \frac{\mu}{2}\|\mathbf{C}-\mathbf{B}\|_F^2 \\ \text{s.t. } \mathbf{C} \in \mathcal{S}_{p,n}, \mathbf{B} \in \Omega \end{aligned}, \tag{4}$$

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, two steps alternative iterative optimization algorithm, see Algorithm 1, can be expolited to solve this optimization problem(4).

Algorithm 1 Alternative iterative algorithm to (4)

Require: Initializing $\mathbf{C}^0$, $\mathbf{B}^0$ and μ^0, $\sigma > 1$;
while not convergence **do**

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

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

$$\mu^{k+1} = \sigma\mu^k$$

end while
Output the hash code $\mathbf{B}$.

However, as above-mentioned, the orthogonal 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 propose two algorithms to efficiently tackle these two subproblems, respectively.

B. Penalty function method to orthogonal constraint optimization subproblem (5)

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

$$\min_{\mathbf{C}} \widetilde{\mathcal{F}}(\mathbf{C},\mathbf{B}) \quad \text{s.t. } \mathbf{C}\mathbf{C}^\top = \mathbf{I}_p, \tag{7}$$

where

$$\widetilde{\mathcal{F}}(\mathbf{C},\mathbf{B}) = \mathcal{F}(\mathbf{C},\mathbf{B}) + \frac{\mu}{2}\|\mathbf{C}-\mathbf{B}\|_F^2.$$

There are several ways to penalize the orthogonal constraints into the objective function and the easiest way is the following penalty function model

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

where β is a penalty parameter. For some special form of objective function, Wen[16] et al have proved that subproblem (7) is equivalent to an unconstrained penalty function model(8) when choosing an appropriate finite large β. In addition, when β is chosen properly, the number of saddle points of (8) is less than that of (7). However, since the entire algorithm has rigorous requirements for parameters selection, thus, solution to this model is often limited by the problem itself.

Another commonly used penalty to solve the problem (7) is the augmented Lagrangian method (ALM), which the augmented Lagrangian penalty function can be written as:

$$\mathcal{L}(\mathbf{C},\Lambda) := \widetilde{\mathcal{F}}(\mathbf{C},\mathbf{B}) - \frac{1}{2}\langle\Lambda, \mathbf{C}\mathbf{C}^\top - \mathbf{I}_p\rangle + \frac{\beta}{4}||\mathbf{C}\mathbf{C}^\top - \mathbf{I}_p||_F^2, \tag{9}$$

where $\Lambda \in \mathsf{R}^{p\times p}$ is called the Lagrangian multiplier corresponding to the orthogonality constraints. Using ALM, problem (7) can be solved by updating the prime variables $\mathbf{C}$ and Lagrange multipliers Λ alternatively.

Inspired by observation that the dual variables Λ enjoy a closed-form formula at each first-order stationary point, a proximal linearized augmented Lagrangian method (PLAM) is established in[31] to solve optimization (9), which apply the symmetrization of this closed-from formula to Λ as the updating rule for the dual variables and a proximal linearized approximation to a merit function is minimized to update the prime variables $\mathbf{C}$. Specifically, the merit function, which is used to evaluate the function value reduction of PLAM, is defined as

$$\mathcal{L}(\mathbf{C}) := \widetilde{\mathcal{F}}(\mathbf{C},\mathbf{B}) - \frac{1}{2}\langle\Lambda(\mathbf{C}), \mathbf{C}\mathbf{C}^\top - \mathbf{I}_p\rangle + \frac{\beta}{4}||\mathbf{C}\mathbf{C}^\top - \mathbf{I}_p||_F^2, \tag{10}$$

where $\Lambda(\mathbf{C}) = \Phi(\nabla\widetilde{\mathcal{F}}(\mathbf{C},\mathbf{B})^\top\mathbf{C})$ and $\Phi : R^{p\times p} \to R^{p\times p}$, $\Phi(M) = \frac{M+M^\top}{2}$ denotes a linear operator that symmetrizes M.

Unfortunately, either ALM or PALM model also has rigorous requirement for parameter β selection; in fact, it is even not clear how to choose the parameter β appropriately in practice. To try to solve this problem, the authors in[31] proposed to impose a column-wise unit sphere constraint to (10), that is the compact Oblique manifold constraint

$$\mathcal{OB}_{p,n} := \{\mathbf{C} \in \mathsf{R}^{p\times n} | \|\mathbf{C}(:,i)\|_2 = 1, i = 1, ..., p\}.$$

The authors in[32] further extend this constraint to a convex set

$$\mathcal{M} := \{\mathbf{C} \in \mathsf{R}^{p\times n} | ||\mathbf{C}||_F \leq K\}, \tag{11}$$

where $K > \sqrt{p}$ 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 $\mathcal{M}$. By imposing this compact constraint on the objective function and without need additional high computational cost, they have also proved that this convex relaxation scheme can effectively reduce the requirement for parameter β selection.

Under these analysis, intuitively, it seems that we may apply this method to solve subproblem (5) by solving the following optimization problem

$$\min_{\mathbf{C}\in\mathcal{M}} \mathcal{L}(\mathbf{C}). \tag{12}$$

However, our numerical experiments show that this method owns high computational complexity to the hash coding problem and the performance is also not as perfect as imagined.

Inspired by thse observations, we propose to solve the following optimization problem that use penalty function formulation of (7) to be objective function and imposed with convex constraint $\mathcal{M}$, i.e.

$$\min_{\mathbf{C}\in\mathcal{M}} \mathcal{L}(\mathbf{C}) := \widetilde{\mathcal{F}}(\mathbf{C}, \mathbf{B}) + \frac{\beta}{4}||\mathbf{C}\mathbf{C}^\top - \mathbf{I}_p||_F^2. \tag{13}$$

The equivalence between the orthogonal constraint optimization subproblem (5) and the corresponding penalty function optimization problem (13) can be established based on **Theorem 2.1** in [16] and **Theorem 3.3** in [32], see more details in these literature. We will discuss algorithm designed for solving (13) in Section IV.

C. 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 $\mathbf{B}$,

$$\begin{aligned} &\mathbf{B}^* = \arg\min_{\mathbf{B}} \mathcal{F}(\overline{\mathbf{C}}, \mathbf{B}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \mathbf{B}\|_F^2 \\ &\text{s.t. } \mathbf{B}\mathbf{1}_n = \mathbf{0}_p, \mathbf{B} \in \{-1, 1\}^{p\times n} \end{aligned}. \tag{14}$$

Generally, there exist two difficulties when we try to solve (14). 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(14) 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.*

From this definition, it can be seen that semi-continuous variable either is 0(zero), thus is discrete, or lie within some continuous nonnegative range, where the semi-continuous lower bound δ must be finite and nonnegative, while the upper bound need not be finite. 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 (14), 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:

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

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

$$x^{k+1} = \text{hard}(y, \lambda) = \begin{cases} y, |y| \geq \lambda \\ 0, |y| < \lambda \end{cases} \tag{15}$$

where y is every element of vector $\mathbf{y} = \mathbf{x}^k - \alpha\nabla_{\mathbf{x}} f(\mathbf{x}^k)$($\nabla_{\mathbf{x}} f(\mathbf{x})$ is the gradient of $f(\mathbf{x})$) 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 (14) 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

$$||\mathbf{b}_i^\top + \mathbf{1}_n||_0 = \frac{n}{2}, \tag{16}$$

where $\mathbf{b}_i$ is the i-th row of binary discrete matrix $\mathbf{B}$. Then, the problem (14) can be reformulated as follow

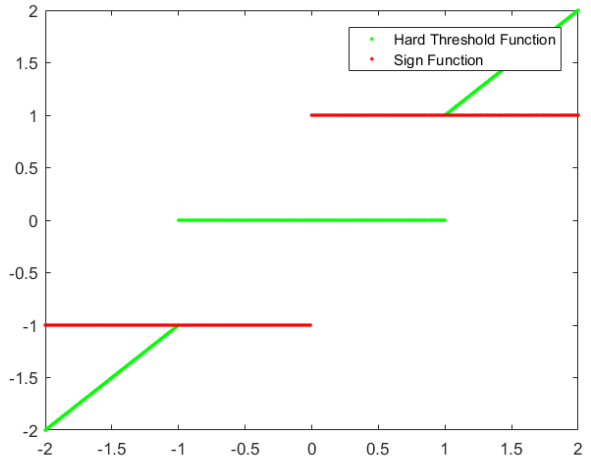


Fig. 1: The image of the hard threshold function and the *sgn* function, where $\lambda = 1$.

$$\begin{aligned} &\mathbf{B}^* = \arg\min_{\mathbf{B}} \mathcal{F}(\overline{\mathbf{C}}, \mathbf{B}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \mathbf{B}\|_F^2 \\ &\text{s.t. } ||\mathbf{b}_i^\top + \mathbf{1}_n||_0 = \frac{n}{2}, i = 1, ...p, \mathbf{B} \in \{-1, 1\}^{p\times n} \end{aligned}. \tag{17}$$

Although equation (16) 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 (14).

One possible algorithm to solve (17) is first to solve a continuous relaxation optimization problem using hard thresholding method directly, i.e. solving the following problem

$$\begin{aligned} &\widetilde{\mathbf{B}}^* = \arg\min_{\widetilde{\mathbf{B}}} \ \mathcal{G}(\widetilde{\mathbf{B}}) := \mathcal{F}(\overline{\mathbf{C}}, \widetilde{\mathbf{B}}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \widetilde{\mathbf{B}}\|_F^2 \\ &\text{s.t. } ||\tilde{\mathbf{b}}_i^\top + \mathbf{1}_n||_0 = \frac{n}{2}, i = 1, ...p \end{aligned}, \tag{18}$$

where $\widetilde{\mathbf{B}}$ is continuous alias for $\mathbf{B}$. Using (15), it can be found that the solution $\widetilde{\mathbf{B}}^* = \text{hard}(\overline{\mathbf{B}}, \lambda)$ to (18) is semi-continuous due to the hard thresholding function with appropriate selected parameter λ, where

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

After that, applying the *sgn* function

$$sgn(x) = \begin{cases} 1, x > 0 \\ -1, x \le 0 \end{cases}$$

to the semi-continuous $\widetilde{\mathbf{B}}^*$ to obtain the binary discrete coding $\mathbf{B}^*$, that is,

$$\mathbf{B}^* = sgn(\text{hard}(\overline{\mathbf{B}}, \lambda)). \tag{19}$$

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, from Fig. 1, when we look at the hard threshold function from the right side of the y-axis, both hard thresholding function and *sgn* function share the property that can convert a continuous variable to a non-continuous variable. Particularly, 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 $sgn(\text{hard}(\cdot))$ may efficiently reduce the quantization error by avoiding the directly discretization to the continuous variables.

However, it is difficult to use the combination operator (19) directly to maintain the balanced constraints, and is hard to deal with both constraints properly and simultaneously, because the performance of *sgn* and hard thresholding function on continuous variable are not completely consistent. To circumvent this problem, we propose a semi-hard thresholding function and a semi-*sgn* function to replace the hard thresholding function and *sgn* function in (19), respectively.

We firstly reformulate (17) 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.

$$\begin{aligned} &\mathbf{Y}^* = \arg\min_{\mathbf{Y}} \mathcal{F}(\overline{\mathbf{C}}, \mathbf{Y}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \mathbf{Y}\|_F^2 \\ &\text{s.t. } \|\mathbf{y}_i\|_0 = \frac{n}{2}, i = 1, ...p, \mathbf{Y} \in \{0, 1\}^{p\times n} \end{aligned}, \tag{20}$$

where $\mathbf{y}_i$ is the $i-$th row of matrix $\mathbf{Y}$. In this case, we can view this optimization model as a *rigid* 2-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.2.(b). However, the solution obtained by (19) 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 result of the gradient descent, is $\gamma > 0$, there may be k negative numbers ($k \le \frac{n}{2}$) whose absolute value is greater than γ. Then by (19) and due to the *sgn* function, the number of -1 in one row of $\mathbf{Y}^*$ will be ($n - \frac{n}{2} + k \le \frac{n}{2}$), as shown in Fig.2.(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 (19) it will be $[-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1]$. Therefore, (19) 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

$$\text{hard}_{semi}(y, \delta) = \begin{cases} y, & y \ge \delta \\ 0, & y < \delta \end{cases}, \tag{21}$$

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. 3 for setting $\delta = -1$. The semi-*sgn* function is defined by

$$sgn_{semi}(x) = \begin{cases} 1, & x \ne 0 \\ -1, & x = 0 \end{cases}. \tag{22}$$

However, this semi-hard thresholding function can not reduce the quantization error efficiently, which the reason

(a) (b)

Fig. 2: (a) is the possible situation that can be solved by (19) (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).

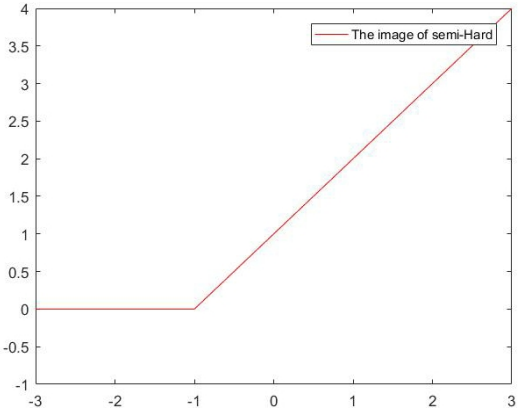


Fig. 3: The image of the semi-hard, when $\delta = -1$.

is $\mathbf{Y} \in \{0,1\}^{p\times n}$ still is a strong discrete constraint to problem (20), thus conflicts with the semi-continuous requirement that we above mentioned. Therefore, we further relax $\mathbf{Y} \in \{0,1\}^{p\times n}$ to an extended semi-continuous matrix and reformulate problem (20) into the following semi-continuous optimization problem

$$\begin{aligned}\mathbf{Y}^{k+1} =& \arg\min_{\mathbf{Y}} \ \mathcal{G}(\mathbf{Y}) := \mathcal{F}(\overline{\mathbf{C}}, \mathbf{Y}) + \frac{\mu}{2}\|\overline{\mathbf{C}} - \mathbf{Y}\|_F^2 \\ &\text{s.t. } \|\mathbf{y}_i\|_0 = \frac{n}{2}, i = 1, ...p, \mathbf{Y} \in \Omega_\delta(0)\end{aligned}, \quad (23)$$

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

$$\Omega_\delta(0) \in \begin{cases} \{\{0\} \cup [\delta, +\infty)\}^{p\times n} & \text{if } \delta \geq 0 \\ \{[\delta, 0) \cup \{0\} \cup (0, +\infty)\}^{p\times n} & \text{if } \delta < 0 \end{cases}, \quad (24)$$

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[52], solution to (23) in one iteration can be expressed by

$$\mathbf{Y}^{k+1} = \text{hard}_{semi}(\overline{\mathbf{Y}}, \delta). \quad (25)$$

And the final hash code can be achieved by applying semi-sgn (22) to solution $\mathbf{Y}^*$ of (23), i.e.

$$\mathbf{B} = sgn_{semi}(\mathbf{Y}^*).$$

Let's look at the example we gave above. The solution obtained by (25) 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(23) can not only be solved efficiently using semi-had thresholding function (25) but also can be solved with less quantization error since the extended semi-continuous relaxation (24) to discrete binary constraints.

D. Equivalence analysis

To analyze whether the problem (1) can be solved in the form of the problem (23), we only need to verify whether the solution obtained by (25) 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. According to Definition 2 of $supp$ and (22) sgn_{semi}, this lemma is very easy to obtain. □

Lemma 1 shows that the sgn_{semi} function (22) 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* (23) *obtained by* (25) *is equivalent to that of the original problem* (1)*.*

Proof. To prove this theorem, we need to prove that the solution of (23) obtained by (25) is the solution of the original problem (1). For convenience, let $\mathbf{B}^*$ be the close-formed solution[36] of the original problem and $\mathbf{B}'$ be the solution obtained by (25). According to the results in paper[16], when μ is chosen properly, it is easy to obtain that (1) is equivalent to (4). With the help of **Theorem 4.2** in [31], we can prove the equivalence for any fixed $\mathbf{B}$, which implies that solution to (14) is equivalent to solution to (1). On the other hand, even though (16) 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 (14). 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 (23) of (14) 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 (24) 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.

IV. Optimization and Convergence

In this section, according to the previous analysis and discussion, we propose an alternative optimization algorithm to solve hash coding problem (4) by reformulating it as follows

$$\begin{aligned}&\min_{\mathbf{C},\mathbf{B}} \mathcal{L}(\mathbf{C},\mathbf{B})\\ &\text{s.t. } \mathbf{C}\in\mathcal{M}, \|\mathbf{b}_i+1\|_0=\frac{n}{2}, i=1,...p, \mathbf{B}+\mathbf{1}\in\Omega_\delta(0)\end{aligned}, \tag{26}$$

where

$$\mathcal{L}(\mathbf{C},\mathbf{B}) = \mathcal{F}(\mathbf{C},\mathbf{B}) + \frac{\beta}{4}||\mathbf{C}\mathbf{C}^\top - \mathbf{I}_p||_F^2 + \frac{\mu}{2}\|\mathbf{C}-\mathbf{B}\|_F^2,$$

and $\mathcal{M}$, $\Omega_\delta(0)$ are defined in (11) and (24), respectively. In the following, we give the detailed algorithm and the convergence analysis of the entire algorithm.

A. Optimization algorithm

*1. **Fix** $\mathbf{B}$, **optimize** $\mathbf{C}$*

Given the kth iterative $\mathbf{B}^k$ that is a semi-continuous matrix and by the analysis in B subsection in Section III, solution of $\mathbf{C}$ in (26) can be obtained by solving the following convex optimization subproblem,

$$\min_{\mathbf{C}\in\mathcal{M}} \mathcal{L}(\mathbf{C},\mathbf{B}^k). \tag{27}$$

As we have analyzed previously, this is a variant of directly penalty function method with a relatively regular manifold constraint $\mathcal{M}$. Notice that the objective is differential with respect to variable $\mathbf{C}$ and the constraint $\mathcal{M}$ is a relatively regular sphere, then the forward-backward method[52] can be used directly to solve it. Firstly, we need to update $\mathbf{C}$ using the gradient descent method,

$$\widetilde{\mathbf{C}}^{k+1} = \mathbf{C}^k - \gamma_k \nabla_{\mathbf{C}}\mathcal{L}(\mathbf{C}^k,\mathbf{B}^k), \tag{28}$$

where $\nabla_{\mathbf{C}}\mathcal{L}(\mathbf{C}^k,\mathbf{B}^k)$ is the gradient with respect to $\mathbf{C}$ at kth iterative result $\mathbf{C}^k$, γ_k is step length parameter. This step is also known as the $forward$ step in framework forward-backward algorithm. Then, in the $backward$ step, we will project the result from (28) on the convex sphere constraint $\mathcal{M}$ shown as follows

$$\mathbf{C}^{k+1} = proj_{\mathcal{M}}(\widetilde{\mathbf{C}}^{k+1}), \tag{29}$$

where

$$proj_{\mathcal{M}}(\widetilde{\mathbf{C}}^{k+1}) := \frac{K}{||\widetilde{\mathbf{C}}^{k+1}||_2}\widetilde{\mathbf{C}}^{k+1}.$$

*2. **Fix** $\mathbf{C}$, **optimize** $\mathbf{B}$*

Suppose that $\mathbf{C}^{k+1}$ from (29) is known, by the analysis in C subsection in Section III, $\mathbf{B}$ in (26) can be achieved by solving the following semi-continuous optimization problem

$$\begin{aligned}&\mathbf{B}^{k+1} = \arg\min_{\mathbf{B}} \mathcal{L}(\mathbf{C}^{k+1},\mathbf{B})\\ &\text{s.t. } \|\mathbf{b}_i+1\|_0=\frac{n}{2}, i=1,...p, \mathbf{B}+\mathbf{1}\in\Omega_\delta(0)\end{aligned}. \tag{30}$$

It should note that the $\mathbf{B}+\mathbf{1}\in\Omega_\delta(0)$ constraint here is only to restrict that $\mathbf{B}+\mathbf{1}$ is an extended semi-continuous variable. Thus, according to the proposed iterative semi-hard thresholding optimization algorithm in Section III, we first need to obtain the gradient descent of objective function at kth iterative point $\mathbf{B}^k$), i.e.

$$\overline{\mathbf{B}} = \mathbf{B}^k - \alpha_k \nabla_{\mathbf{B}}\mathcal{L}(\mathbf{C}^{k+1},\mathbf{B}^k), \tag{31}$$

where α_k is the step length. Then, using the semi-continuous threshold operator (25), we have

$$\mathbf{B}^{k+1} = \text{hard}_{semi}(\overline{\mathbf{B}},\delta), \tag{32}$$

where δ is the $\frac{n}{2}$-th largest value of a row in $\overline{\mathbf{B}}$. The reason why we do not use $\overline{\mathbf{B}}+\mathbf{1}$ as the input in (32) is that the output of semi-hard thresholding process to $\overline{\mathbf{B}}+\mathbf{1}$ and $\mathbf{B}$ actually is similar, which we have shown in the proof of **Theorem** 1.

*3. **Update** μ*

For continuous variable $\mathbf{C}$ and semi-continuous variable $\mathbf{B}$, we hope to continuously reduce the gap through an iterative methods. This step is mainly achieved through iterative minimizing $\frac{\mu}{2}\|\mathbf{C}-\mathbf{B}\|_F^2$, where it is required the penalty parameter $\mu\to\infty$ theoretically. However, in practice, it is hard to satisfy this condition and actually is unnecessary. Indeed, we just only approach infinity as much as possible by increasing the parameter at every iteration, i.e. to update μ using the following scheme in our experiments

$$\mu^{k+1} = \sigma\mu^k, \tag{33}$$

where $\sigma > 1$. In addition, parameter β in objective function of (26) is also hard to determine, even though we have analyzed its role in (8). In our experiments, we empirically set parameter β to be a different constant according to different dataset, and in this case the algorithm shows convergence. Theoretical analysis of the setting of these paprameters will be considered in the future work.

At last, repeating these three steps until algorithm convergence, then the final hash code can be obtained by using the following formula,

$$Hash = sgn_{semi}(\mathbf{B}^*), \tag{34}$$

where $\mathbf{B}^*$ is the optimal solution to (26). We summarize all the steps in **Algorithm** 2.

B. Convergence of the proposed algorithm

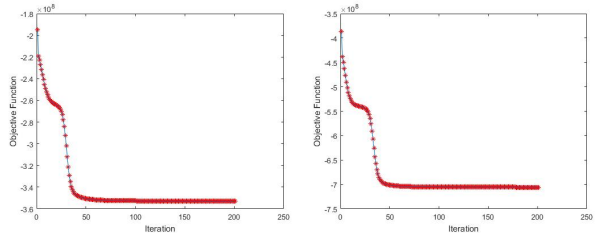
From **Algorithm 2**, we can prove that objective function $\mathcal{L}(\mathbf{C}^k,\mathbf{B}^k)$ non-increase monotonously for each iterative points, i.e. satisfies

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

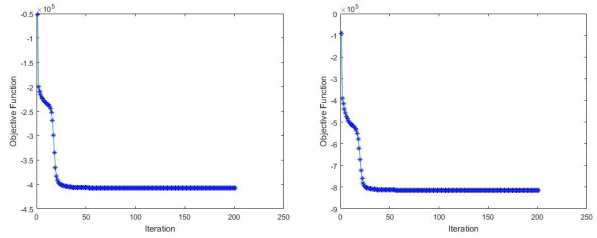
Algorithm 2 The PFSCT algorithm

Require: Randomly choose C_0 on Stiefel manifold and B_0 satisfying the constraint, set $K > \sqrt{p}$, $\sigma > 1$, β is a constant and $k = 0$
while not convergence **do**
(C-Step)
Compute $\widetilde{C}^{k+1}$ by (28)
if $||\widetilde{C}^{k+1}||_F > K$ **then**
Compute C^{k+1} by (29)
else
$C^{k+1} = \widetilde{C}^{k+1}$
end if
(B-Step)
Update B^{k+1} by (32)
(μ-Step)
$\mu^{k+1} = \sigma\mu^k$
end while
Get the hash code by (34)

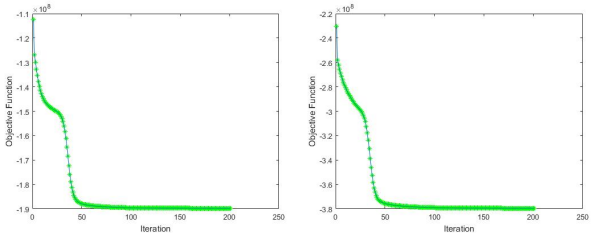
Under some appropriate assumptions to objective function, for example, is low-bounded and proper, we may prove the proposed alternative iterative algorithm is convergence theoretically. And, in the following section, the extensive experiments also numerically show that our proposed algorithm is convergence.



(a)Cifar 10 with 16 bits (b)Cifar10 with 32 bits



(c)NUS-WIDE with 16 bits (d)NUS-WIDE with 32 bits



(c)SUN397 with 16 bits (d)SUN397 with 32 bits

Fig. 4: The objective function value of unsupervised hash model solved by **Algorithm 2**.

V. Numerical Experiment

In this section, we conduct experiments using the proposed PFSCT based on the unsupervised spectral graph hash model[51] and the supervised strongly constrained discrete hashing model[36], 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[51] and DPLM[11] for solving the unsupervised model[51], also comparing it with the kernelized algorithm of SCDH(SCDH$_k$)[36] and DPLM[11] for solving the supervised strongly constrained discrete hashing model[36]. 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[2], PCAH[50], DSH[33] and supervised methods KSH[34], SDH[47], FastH[35]. 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.

A. Datasets

Cifar 10 is labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, etc[51], which consists of 10 different categories of (of size $32\times32\times3$) 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[52]. 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[53] 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.

B. 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

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

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.

C. Parameter Setting

The 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 2** for unsupervised objective function from[51] and $\beta = 1$, $\mu = 5$, $\lambda = 10$ and $\mu = 5$ of **Algorithm 3** for supervised objective function from [36].

D. 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[51] as an example, and compare the performance of PFSCT approache with the recently developed DPLM method [11] and the classical SH method[51]. 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.

As being discussed in Section IV, this unsupervised hash problem can be rewritten as follow:

$$\min_{\mathbf{C},\mathbf{B}} \mathcal{F}(\mathbf{C}, \mathbf{B})$$

$$s.t. \quad \mathbf{B}\mathbf{1}_n = 0, \mathbf{B} \in \Omega_\delta(0), \mathbf{C} \in \mathsf{R}^{p\times n},$$

where

$$\mathcal{F}(\mathbf{C}, \mathbf{B}) = \frac{1}{2}tr(\mathbf{C}\mathbf{L}\mathbf{C}^\top) + \frac{\beta}{4}||\mathbf{C}\mathbf{C}^\top - \mathbf{I}_p||_F^2 + \frac{\mu}{2}||\mathbf{C} - \mathbf{B}||_F^2$$

and $\Omega_\delta(0)$ is the extended semi-continuous domain defined by(24), $\mathbf{B}$ is a semi-continuous matrix and $\mathbf{C}$ is its continuous relaxation. By **Algorithm 2**, 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

$$\min_{\mathcal{H}} \|\mathbf{B} - \mathcal{H}\mathbf{X}\|_F^2. \tag{35}$$

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

TABLE I: THE mAP SCORES ON CIFAR10 BASHED ON UNSUPERVISED HASH MODEL.(THE BEST RESULTS ARE IN BOLD)

Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
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 II: THE mAP SCORES ON NUS-WIDE BASHED ON UNSUPERVISED HASH MODEL.(THE BEST RESULTS ARE IN BOLD)

Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
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

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. 4, 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

TABLE III: THE mAP SCORES ON SUN397 BASHED ON UNSUPERVISED HASH MODEL.(THE BEST RESULTS ARE IN BOLD)

Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
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

hand, the mAP values achieved from PFSCT, SH and DPLM algorithms respectively are shown in Table I, II and III. 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.

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[54].

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[36], 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.

E. 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 [11] and the SCDH$_k$[36]. 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 III as the following optimization problem

$$\min_{\mathbf{C},\mathbf{B},\mathbf{A}} \mathcal{F}(\mathbf{C},\mathbf{B},\mathbf{A})$$
$$s.t. \mathbf{B} \in \Omega_\delta(0), \mathbf{B}^\top \mathbf{1}_n = 0_r, \mathbf{C} \in \mathsf{R}^{n\times r},$$

where

$$\mathcal{F}(\mathbf{C},\mathbf{B},\mathbf{A}) = ||r\cdot\mathbf{S} - \mathbf{B}\mathbf{C}^\top||_F^2 + \lambda||\mathcal{K}_\mathcal{Q}\mathbf{A} - \mathbf{B}||_F^2 + \alpha||\mathbf{B}-\mathbf{C}||_F^2 + \beta||\mathbf{A}||_F^2 + \mu||\mathbf{C}^\top\mathbf{C} - n\cdot\mathbf{I}_r||_F^2$$

and kernel matrix $\mathcal{K}_\mathcal{Q}$ is constructed using the original data and its corresponding anchors, see more details in [36].

Compared with solving unsupervised hash model (2) that does not involve the hash function, for solving this supervised model, we need to add an extra step to train hash function, i.e. to train $\mathbf{A}$ by solving the following optimization problem

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

where $\gamma = \beta/\lambda$. This problem is sample and the closed solution is

$$\mathbf{A} = (\mathcal{K}_\mathcal{Q}{}^\top\mathcal{K}_\mathcal{Q} + \gamma\mathbf{I}_\mathbf{M})^{-1}\mathcal{K}_\mathcal{Q}{}^\top\mathbf{B}. \tag{36}$$

The matrix variables $\mathbf{B}$ and $\mathbf{C}$ can still be solved by the established **Algorithm 2**. For the sake of integrity of this paper, we illustrate the detailed algorithm in **Algorithm 3**.

Algorithm 3 The PFSCT algorithm to supervised hashing model

Require: Randomly choose C_0 on Stiefel manifold and B_0 satisfying the constraint, set $K > \sqrt{r}$ and $k = 0$
while not convergence **do**
(A-Step)
Compute $\mathcal{A}$ by (36)
Update $\tilde{C}$, $\tilde{B}$ and μ follow **Algorithm** 2
end while
Get the hash code by (34)

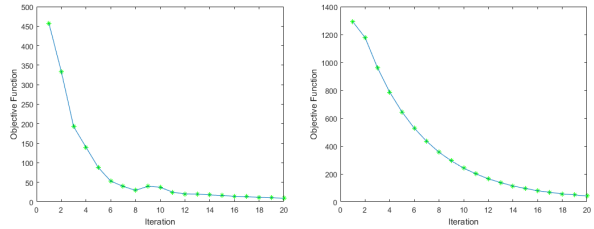
Based on the same dataset as being used in unsupervised hash code model, we have conducted supervised hash numerical experiments using **Algorithm 3**. The value of objective function decay very fast seen from Fig. 5 for the 8, 16 bits case, which indicates convergence of algorithm.

TABLE IV: THE mAP SCORES ON CIFAR10 BASHED ON SUPERVISED HASH MODEL.(THE BEST RESULTS ARE IN BOLD)

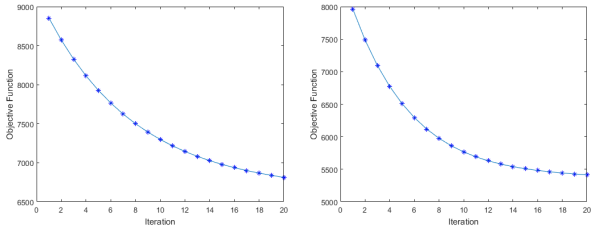
Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
SCDH$_K$	0.6506	0.6509	**0.6526**	**0.6537**
DPLM	0.6508	0.6509	0.6510	0.6513
PFSCT	**0.6508**	**0.6512**	0.6514	0.6516

TABLE V: THE mAP SCORES ON NUS-WIDE BASHED ON SUPERVISED HASH MODEL.(THE BEST RESULTS ARE IN BOLD)

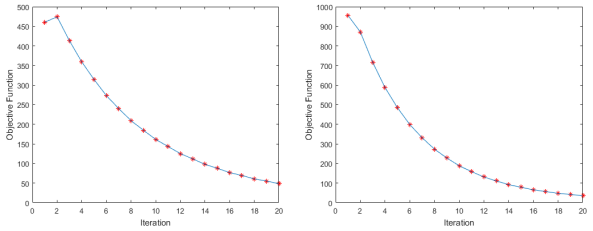
Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
SCDH$_K$	0.6680	0.6682	**0.6712**	**0.6727**
DPLM	0.6679	0.6680	0.6682	0.6683
PFSCT	**0.6681**	**0.6690**	0.6689	0.6683



(a)Cifar 10 with 8 bits (b)Cifar10 with 16 bits



(c)NUS-WIDE with 8 bits (d)NUS-WIDE with 16 bits



(c)SUN397 with 8 bits (d)SUN397 with 16 bits

Fig. 5: The objective function value of supervised hash model solved by **Algorithm 3**.

TABLE VI: THE mAP SCORES ON SUN397 BASHED ON SUPERVISED HASH MODEL.(THE BEST RESULTS ARE IN BOLD)

Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
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

The mAP values achieved from PFSCT, SCDH$_k$ and DPLM algorithms respectively are shown in Table IV, V and VI. Unsurprisingly, compared with the 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, this is because

F. 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 3** 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 VII: THE mAP SCORES ON CIFAR10 HAND-CRAFTED DATASET BASED ON SUPERVISED HASH MODEL.(THE BEST RESULTS ARE IN BOLD)

Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
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 VII, it can be seen that mAP values using feature dataset significantly outperform the mAP values that using original image dataset shown in Table IV. 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.

G. 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[2], PCAH[50], DSH[33] are unsupervised methods, while the KSH[34], SDH[47], FastH[35] 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.

Unsurprisingly, the supervised models are superior to the unsupervised method as shown in Table VIII and IX, 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.

TABLE VIII: THE mAP SCORES OF BASELINE HASHING METHODS ON CIFAR10.(THE BEST RESULTS ARE IN BOLD)

Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
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 for SCDH model	**0.6508**	**0.6512**	0.6514	0.6516

TABLE IX: THE mAP SCORES OF BASELINE HASHING METHODS ON NUS-WIDE.(THE BEST RESULTS ARE IN BOLD)

Method/ Length	8bits mAP score	16bits mAP score	32bits mAP score	64bits mAP score
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 for SCDH model	**0.6681**	**0.6690**	0.6689	0.6683

VI. Conclusion And Future

In this article, we propose a new method to deal with the three constraints in the Hash problem. We adopted the method of dealing with the orthogonal constraint first, through a relatively inaccurate penalty and a precise solution technique, to avoid the related operations of eigenvalue decomposition of large matrices, and obtain a acceptable exact solution with lower computational complexity. In the processing of the next two constraints, we successfully introduced semi-continuous variables to reduce the quantization error, combined the characteristics of the ℓ_0 norm with the balance constraint, and gave a threshold function to solve, thereby obtained an accurate solution with a small quantization error. It is verified through experiments that our solution method is comparability.

For the algorithm we proposed, there are still areas that can be improved. The first is about the proof of equivalence. We can only theoretically give when the objective function is a special form, that is, the form of Laplacian feature extraction (SH). Therefore, how to extend the proof of equivalence to a more general form is a question worth considering. Secondly, we give an algorithm framework, under this framework, the subproblem solving method can be improved, and other algorithms with faster solving speed and higher solving accuracy can be introduced. Finally, although we have given a proof of the equivalence of the SH problem and conducted experiments, it is a pity that we still have not solved the problem of calculating the Laplacian matrix of the spectral Hash problem. We believe that the points we mentioned above are worthy of further discussion and have very important practical and economic significance.

References

[1] J. Wang, T. Zhang, J. Song, N. Sebe and H. T. Shen, "A Survey on Learning to Hash," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 769-790, 1 April 2018.

[2] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in Proc. Int. Conf. Very Large Databases, 1999, pp. 518-C529.

[3] B. Kulis, P. Jain and K. Grauman, "Fast Similarity Search for Learned Metrics," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 12, pp. 2143-2157, Dec. 2009.

[4] Y. Yang, Z. Zha, Y. Gao, X. Zhu and T. Chua, "Exploiting Web Images for Semantic Video Indexing Via Robust Sample-Specific Loss," in IEEE Transactions on Multimedia, vol. 16, no. 6, pp. 1677-1689, Oct. 2014.

[5] R. Xia, Y. Pan, H. Lai, C. Liu, S. Yan, "Supervised hashing for image retrieval via image representation learning," in Proceedings of the National Conference on Artificial Intelligence, no. 3, pp. 2156-2162, 2014.

[6] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, Accurate Detection of 100,000 Object Classes on a Single Machine," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[7] K. Q. Weinberger, A. Dasgupta, J. Langford, A. J. Smola, and J. Attenberg, "Feature Hashing for Large Scale Multitask Learning," in Proc.ICML, 2009.

[8] X. Liu, J. He, C. Deng and B. Lang, "Collaborative Hashing," in IEEE Conference on Computer Vision and Pattern Recognition 2014, Columbus, OH, 2014, pp. 2147-2154.

[9] Y. Li, S. Wang, Q. Pan, H. Peng,T. Yang, and E. Cambria, "Learning binary codes with neural collaborative filtering for efficient recommendation systems," in Knowledge-Based systems, vol. 172, pp. 64-75, 2019.

[10] P.-A. Absil, R. Mahony, and R. Sepulchre, "Optimization Algorithms on Matrix Manifolds," in Princeton University Press, Princeton, NJ.

[11] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen and D. Tao, "A Fast Optimization Method for General Binary Code Learning," in IEEE Transactions on Image Processing, vol. 25, no. 12, pp. 5610-5621, Dec. 2016.

[12] K. Zhao, H. Lu, Y. He, and S. Feng, "Locality Preserving Discriminative Hashing," in Proceedings of the 22nd ACM international conference on Multimedia (MM '14), pp. 1089–1092.

[13] J. Hu , X. Liu, Z. Wen and Y. Yuan. "A Brief Introduction to Manifold Optimization," in J. Oper. Res. Soc. China, vol. 8, pp. 199-248, 2020.

[14] Z. Wen, W. Yin. "A feasible method for optimization with orthogonality constraints," in Mathematical Programming. vol. 142, pp. 397-434, 2013.

[15] B. Gao, X. Liu, X. Chen and Y. Yuan, "A New First-Order Algorithmic Framework for Optimization Problems with Orthogonality Constraints." in SIAM Journal on Optimization, vol. 28, no. 1, pp. 302-332, 2018.

[16] Z. Wen, C. Yang, X. Liu, and Y. Zhang, "Trace-penalty minimization for large-scale eigenspace computation," in J. Sci. Comput. vol. 66, pp. 1175–1203, 2016.

[17] J. Song, Y. Yang, Z. Huang, H.T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," in MM'11 - Proceedings of the 2011 ACM Multimedia Conference and Co-Located Workshops, pp. 423-432, 2011.

[18] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10), pp. 18-25, 2010.

[19] J. Wang, S. Kumar and S. Chang, "Semi-supervised hashing for scalable image retrieval," in 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, 2010, pp. 3424-3431.

[20] B. Wu and B. Ghanem, "ℓ_p-Box ADMM: A Versatile Framework for Integer Programming," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 7, pp. 1695-1708, 1 July 2019.

[21] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang and H. T. Shen, "Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 12, pp. 3034-3044, 1 Dec. 2018.

[22] M. Á. Carreira-Perpiñán and R. Raziperchikolaei, "Hashing with binary autoencoders," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 557-566.

[23] J. Lu, V. E. Liong and J. Zhou, "Deep Hashing for Scalable Image Search," in IEEE Transactions on Image Processing, vol. 26, no. 5, pp. 2352-2367, May 2017.

[24] D. Wang, P. Cui, M. Ou and W. Zhu, "Learning Compact Hash Codes for Multimodal Representations Using Orthogonal Deep Structure," in IEEE Transactions on Multimedia, vol. 17, no. 9, pp. 1404-1416, Sept. 2015.
[25] D. Hu, F. Nie and X. Li, "Deep Binary Reconstruction for Cross-Modal Hashing," in IEEE Transactions on Multimedia, vol. 21, no. 4, pp. 973-985, April 2019.
[26] Z. Cao, M. Long, J. Wang and P. S. Yu, "HashNet: Deep Learning to Hash by Continuation," in 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 5609-5618.
[27] X. Luo, L. Nie, X. He, Y. Wu, Z. Chen, and X. Xu, "Fast Scalable Supervised Hashing," in The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18). pp. 735–744.
[28] X. Luo, P. Zhang, Z. Huang, L. Nie and X. Xu, "Discrete Hashing With Multiple Supervision," in IEEE Transactions on Image Processing, vol. 28, no. 6, pp. 2962-2975, June 2019.
[29] S. Zhang, J. Li, M. Jiang, P. Yuan and B. Zhang, "Scalable Discrete Supervised Multimedia Hash Learning With Clustering," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 10, pp. 2716-2729, Oct. 2018.
[30] R. Yang, "New Results on some Quadratic Programming Problems." Order No. 3614752, University of Illinois at Urbana-Champaign, Ann Arbor, 2013.
[31] B. Gao, X. Liu, Y. Yuan, "Parallelizable Algorithms for Optimization Problems with Orthogonality Constraints," in SIAM Journal on entific Computing, vol. 41, no. 3, pp. 1949-1983, 2019.
[32] N. Xiao, X. Liu, Y. Yuan, "A Class of Smooth Exact Penalty Function Methods for Optimization Problems with Orthogonality Constraints," in Optimization Methods and Software, pp. 1-37, 2020.
[33] Z. Jin, C. Li, Y. Lin and D. Cai, "Density Sensitive Hashing," in IEEE Transactions on Cybernetics, vol. 44, no. 8, pp. 1362-1371, Aug. 2014.
[34] W. Liu, J. Wang, R. Ji, Y. Jiang and S. Chang, "Supervised hashing with kernels," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, 2012, pp. 2074-2081.
[35] G. Lin, C. Shen, Q. Shi, A. van den Hengel and D. Suter, "Fast Supervised Hashing with Decision Trees for High-Dimensional Data," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1971-1978.
[36] Y. Chen, Z. Tian, H. Zhang, J. Wang and D. Zhang, "Strongly Constrained Discrete Hashing," in IEEE Transactions on Image Processing, vol. 29, pp. 3596-3611, 2020.
[37] W. Liu, J. Wang, S. Kumar, and S. F. Chang, "Hashing with graphs," in Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11), pp. 1–8, 2011.
[38] W. Liu, C. Mu, S. Kumar, and S. F. Chang, "Discrete graph hashing," in Advances in Neural Information Processing Systems, vol. 4, pp. 3419-3427, 2014.
[39] S. Foucart, "Hard Thresholding Pursuit: An Algorithm for Compressive Sensing," in SIAM Journal on Numerical Analysis, vol. 49, no. 5/6, pp. 2543–2563, 2011.
[40] J. Nocedal, and S. J. Wright, "Numerical optimization. 2nd ed," Springer Science+Business Media, LLC.:USA, pp. 304-321, 2006.
[41] J. H. Manton, "Optimization algorithms exploiting unitary constraints," in IEEE Transactions on Signal Processing, vol. 50, no. 3, pp. 635-650, March 2002.
[42] T. E. Abrudan, J. Eriksson and V. Koivunen, "Steepest Descent Algorithms for Optimization Under Unitary Matrix Constraint," in IEEE Transactions on Signal Processing, vol. 56, no. 3, pp. 1134-1147, March 2008.
[43] R. Lai, S. Osher, "A splitting method for orthogonality constrained problems," in J. Scientific Computing, vol. 58, no. 2, pp. 431-449, 2014.
[44] J. Hu, A. Milzarek, Z. Wen, and Y. Yuan, "Adaptive quadratically regularized newton method for riemannian optimization," in SIAM Journal on Matrix Analysis and Applications, vol. 39, no. 3, pp. 1181-1207, 2018.
[45] A. Edelman, T. A. Arias, and S. T, Smith, "The geometry of algorithms with orthogonality constraints," in SIAM Journal on Matrix Analysis and Applications, vol. 20, no. 2, pp. 303-353, 1999.
[46] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst, "Geometric Deep Learning: Going beyond Euclidean data," in IEEE Signal Processing Magazine, vol. 34, no. 4, pp. 18-42, July 2017.
[47] F. Shen, C. Shen, W. Liu and H. T. Shen, "Supervised Discrete Hashing," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 37-45.
[48] T. Do, K. Le, T. Hoang, H. Le, T. V. Nguyen and N. Cheung, "Simultaneous Feature Aggregating and Hashing for Compact Binary Code Learning," in IEEE Transactions on Image Processing, vol. 28, no. 10, pp. 4954-4969, Oct. 2019.
[49] B. Kulis, T. Darrell, "Learning to hash with binary reconstructive embeddings," in Proceedings of the 22nd International Conference on Neural Information Processing Systems (NIPS'09), pp. 1042–1050, 2009.
[50] Y. Gong, S. Lazebnik, A. Gordo and F. Perronnin, "Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 12, pp. 2916-2929, Dec. 2013.
[51] Y. Weiss, A. Torralba, R. Fergus, "Spectral hashing," in Proceedings of the 21st International Conference on Neural Information Processing Systems," pp. 1753–1760, 2008.
[52] H. Attouch, J. Bolte, B. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods," in Mathematical Programming. vol. 137.
[53] L. Havrlant, V. Kreinovich. "A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation)." in International Journal of General Systems, 2017.
[54] R. Talmon, I. Cohen, S. Gannot and R. R. Coifman, "Diffusion Maps for Signal Processing: A Deeper Look at Manifold-Learning Techniques Based on Kernels and Graphs," in IEEE Signal Processing Magazine, vol. 30, no. 4, pp. 75-86, July 2013.

Qian Chen received the B.S. degree from the University of Science and Technology Beijing, Beijing, China, in 2019, where he is currently pursuing the master degree. His research interests include multimedia retrieval and indexing and computer vision.

Zhengwei Shen is an associate professor in school of mathematics and physics, the University of Science and Technology Beijing, China. He received his M.S. degree in applied mathematics from the University of Science and Technology Beijing, China, in 2004 and his Ph.D. in applied mathematics from Beihang University, Beijing, China, in 2013. His research interests include wavelet analysis, image processing, machine learning and nonlinear optimization.

Zhe Chen is currently an undergraduate in school of mathematics and physics, the University of Science and Technology Beijing, China. His current research direction is hash coding. His research interests also include the cross direction of physics, statics and optimization.