Pattern Recognition Letters 117 (2019) 169–178

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Optimized projection for hashing

Chaoqun Chu[a], Dahan Gong[a], Kai Chen[a], Yuchen Guo[a,*], Jungong Han[b], Guiguang Ding[a,*]

[a] School of Software, Tsinghua University, Beijing 100084, China
[b] School of Computing and Communications, Lancaster University, Lancaster LA1 4YW, UK

ARTICLE INFO

Article history:
Available online 19 April 2018

MSC:
41A05
41A10
65D05
65D17

Keywords:
Hashing
Quantization
Algorithm

ABSTRACT

Hashing, which seeks for binary codes to represent data, has drawn increasing research interest in recent years. Most existing Hashing methods follow a projection-quantization framework which first *projects* high-dimensional data into compact low-dimensional space and then *quantifies* the compact data into binary codes. The projection step plays a key role in Hashing and academia has paid considerable attention to it. Previous works have proven that a good projection should simultaneously 1) preserve important information in original data, and 2) lead to compact representation with low quantization error. However, they adopted a greedy two-step strategy to consider the above two properties *separately*. In this paper, we empirically show that such a two-step strategy will result in a sub-optimal solution because the optimal solution to 1) limits the feasible set for the solution to 2). We put forward a novel projection learning method for Hashing, dubbed *Optimized Projection* (OPH). Specifically, we propose to learn the projection in a unified formulation which can find a good *trade-off* such that the *overall* performance can be optimized. A general framework is given such that OPH can be incorporated with different Hashing methods for different situations. We also introduce an effective gradient-based optimization algorithm for OPH. We carried out extensive experiments for Hashing-based Approximate Nearest Neighbor search and Content-based Data Retrieval on six benchmark datasets. The results show that OPH significantly outperforms several state-of-the-art related Hashing methods.

1. Introduction

The demand for effective indexing structures emerged recently which can perform Approximate Nearest Neighbor (ANN) search efficiently given a large-scale database. One of the best known structures is tree [7] providing logarithmic searching complexity. But tree-based structure may reduce to exhaustive linear search given high-dimensional data [8] which is more common in real world. Hashing, which represents data by binary codes, can effectively cope with such problem. For example, we just need about 1GB memory to load 32 million points with each point represented by 256 bits and performing ANN search needs less than 1 s as only simple bit operations are required to compute Hamming distance [16]. Due to its low storage cost and very high retrieval efficiency, in recent decade Hashing has drawn increasing interest from both academia and industry.

Locality Sensitive Hashing (LSH) [1] is one of the most celebrated models. It adopts random linear projections to map original feature vector to binary codes. Such coding method is quite fast. But in practice, long Hashcodes are required to achieve satisfactory performance because it is data-independent [51]. To tackle this problem, several machine learning techniques have been adopted to design effective and compact Hashcodes, such as Principle Components Analysis, Manifold Learning, Semi-supervised Learning, and Restricted Boltzmann Machine, which respectively lead to PCA Hashing [19], Spectral Hashing [46], and Semi-supervised Hashing [45]. Such deta-dependent Hashing methods can exploit important information hidden in the original features, like global Euclidean distance, local manifold structure, and etc.

Producing binary codes directly from original features is difficult in most cases [46]. Hence, most existing works follow a projection-quantization framework [6,10,15,19,33,34]. Firstly, the (high-dimensional) original features are projected into a low-dimensional compact space whose dimensionality is always equal to the target Hashcode length by a real-value projection function. Secondly, the real-value compact representation is quantified into binary codes by, in most cases, thresholding. A flowchart of such framework is illustrated in Fig. 1 Up. Despite the recent emerging research on quantization [11,16,20,30,39], most researchers have paid and are paying more attention to the projection step [3,19,23,32,45,46,50] because this lays the foundation for

* Corresponding authors.
E-mail addresses: guo-yc13@mails.tsinghua.edu.cn (Y. Guo), dinggg@tsinghua.edu.cn (G. Ding).

https://doi.org/10.1016/j.patrec.2018.04.027
0167-8655/

Hashing and more effective projection can always lead to better result [10].

Observed from literatures, an effective projection should satisfy the following properties simultaneously: 1) preserving important information in original features, such as global Euclidean [19], local manifold structure [34,46], or pair-wise label information [12,31,45], etc.; and 2) leading to projected data with low quantization error which occurs when mapping real-value features to binary codes [10]. Previous works mostly focus on the first property. However, recent studies have demonstrated that the second property is also quite important and optimization aiming at it can result in much better performance [10,49]. Thus they propose to adopt an extra adjustment (a rotation is always utilized) *after the initial projection* to re-project the data for better result, as illustrated in Fig. 1 Down. However, this is a two-step strategy which considers above two properties *separately*. Meanwhile, the second step is limited by the result of initial projection and it only finds the sub-optimal solution.

It is intuitive and straightforward to raise a question: can combining the two steps in projection learning together lead to better result? This paper empirically studies it and obtains a *positive* answer. Motivated by this observation, in this paper, we propose a novel projection learning method for Hashing, referred to as *Optimized Projection* (OPH). Besides, we also make the following contributions in this paper:

- A unified formulation for Hashing projection learning is put forward which can find a good trade-off between preserving information and minimizing quantization error, such that overall performance can be optimized.
- We give a general learning framework for OPH such that it can be incorporated with different Hashing methods based on specific situations. For example, when global Euclidean information is important, we can combine OPH with PCA, while Spectral is adopted if we concern more for the local manifold structure of data.
- For the orthogonality-constrained optimization problem of OPH, we also put forward an effective iterative learning algorithm based on the gradient flow method.
- We carry out extensive experiments for Approximate Nearest Neighbor (ANN) search and Content-based Data (image and text) Retrieval (CBDR) based on Hashing on several benchmark datasets. Experimental results validate the effectiveness of OPH compared with several state-of-the-art related Hashing methods.

2. Observation and motivation

2.1. Problem and notation

Given a set of training data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times d}$, where n is the number of samples and d is the dimensionality of original features, we want to learn a Hashing function h which can produce binary codes $\mathbf{B} = h(\mathbf{X}) = [\mathbf{b}_1, \ldots, \mathbf{b}_n]^T \in \{-1, 1\}^{n\times k}$ for each sample, which are termed as Hashcodes, where k is the length of Hashcodes. Generally, we require the Hashcodes to be balanced ($\mathbf{1}_n\mathbf{B} = \mathbf{0}$) and uncorrelated ($\mathbf{B}^T\mathbf{B} = n\mathbf{I}_k$). Designing h directly is difficult and sometimes NP-hard [46], so we can adopt a projection-quantization strategy. Specifically, we can find a projection matrix $\mathbf{P} \in \mathbb{R}^{d\times k}$, and let $h(\mathbf{x}) = \text{sign}(\mathbf{x}\mathbf{P})$. Here $\text{sign}(x) = 1$ if $x \geq 0$ or -1 otherwise. The sign function is widely used in previous works [10,19,34,45,46] for quantization. Of course we can adopt more complicated quantization functions [16,20]. But this is not the focus of this and some related previous papers thus we still use sign function for fair comparison. Without loss of generality, in this paper we assume the data to be zero centered, i.e., $\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0}$. Consequently, the projected data $\mathbf{Y} = \mathbf{X}\mathbf{P}$ is zero centered as well.

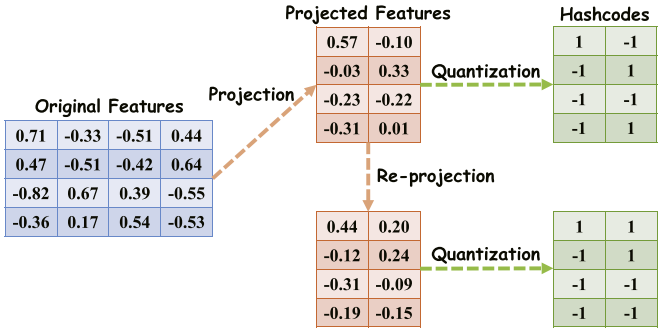


Fig. 1. The flowchart of Hashing. Up) The projection-quantization framework. Down) An extra re-projection is adopted for better result.

2.2. An observation on PCA projection

PCA projection has been widely utilized in several Hashing methods [10,16,21] as the initial projection. Here we analyze the property of PCA for Hashing. In PCA projection, the global Euclidean structure is expected to be preserved by *minimizing the reconstruction error* under a linear orthogonal projection $\mathbf{P}$. Specifically, such projection matrix can be learned by the optimization problem as below

$$\min_{\mathbf{P},\mathbf{Y}} \ \|\mathbf{X} - \mathbf{Y}\mathbf{P}^T\|_F^2, \ s.t. \ \mathbf{P}^T\mathbf{P} = \mathbf{I}_k, \mathbf{Y} = \mathbf{X}\mathbf{P} \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of matrix. Based on the constraints, it is easy to verify that the projection error (reconstruction error mentioned above) can be computed as

$$e_p = \|\mathbf{X} - \mathbf{Y}\mathbf{P}^T\|_F^2 = \|\mathbf{X}\|_F^2 - \|\mathbf{X}\mathbf{P}\|_F^2 \tag{2}$$

Since $\|\mathbf{X}\|_F^2$ is a constant, by substituting Eq. (2) into problem (1), we obtain the final objective function of PCA,

$$\max_{\mathbf{P}} \ \|\mathbf{X}\mathbf{P}\|_F^2, \ s.t. \ \mathbf{P}^T\mathbf{P} = \mathbf{I}_k \tag{3}$$

which can be regard as maximizing the total variance of projected data because the data is zero centered. Problem (3) can be efficiently optimized by eigenvalue decomposition.

The projection $\mathbf{P}$ obtained above can preserve the global Euclidean structure, i.e., it satisfies the first property. Now let us consider the second property, minimizing quantization error. Given real-value data $\mathbf{Y}$, and corresponding binary codes $\mathbf{B} = \text{sign}(\mathbf{Y})$, the quantization error caused by sign function is defined as the distance between them as follows,

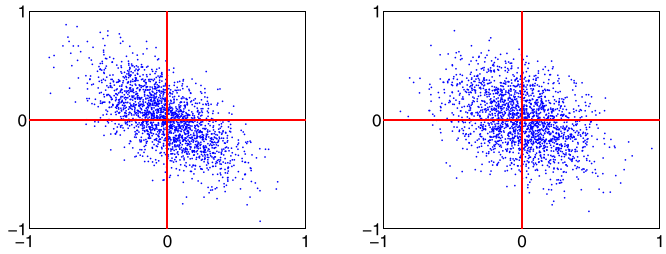
$$e_q = \|\mathbf{B} - \mathbf{Y}\|_F^2 = \|\mathbf{B}\|_F^2 + \|\mathbf{Y}\|_F^2 - 2\|\mathbf{Y}\|_1 \tag{4}$$

where $\|\mathbf{Y}\|_1 = \sum_{i,j} |y_{ij}|$. We have the last term above because $b_{ij} = \text{sign}(y_{ij}) \Rightarrow b_{ij}y_{ij} = |y_{ij}|$. Since we have $\|\mathbf{B}\|_F^2 = nk$ is a constant and $\mathbf{Y} = \mathbf{X}\mathbf{P}$, the projection which can minimize the quantization error can be learned as below

$$\max_{\mathbf{P}} \ 2\|\mathbf{X}\mathbf{P}\|_1 - \|\mathbf{X}\mathbf{P}\|_F^2, \ s.t. \ \mathbf{P}^T\mathbf{P} = \mathbf{I}_k \tag{5}$$

Comparing problem (3) to (5), we can obtain an interesting observation: directly maximizing (5) longs for smaller $\|\mathbf{X}\mathbf{P}\|_F^2$ which is against maximizing (3). In most situations, the respective optimal solutions to (3) and (5) are different.

Above we show the direct connection and contradiction between minimizing projection error and quantization error. However, such relationship between above two errors is ignored in previous works [10,21]. Consequently, a greedy two-step strategy considering two properties *separately* is widely adopted. One representative work is Iterative Quantization (ITQ) [10]. Specifically, they



(a) Small e_p, large e_q (b) Large e_p, small e_q

Fig. 2. Toy data. The value of $\|\mathbf{Y}\|_F^2$ for left (right) is 264.4 (261.8). On the other hand, the overall quantization error for left (right) is 2642.5 (2628.3).

first optimize problem (3) and obtain an initial projection $\mathbf{P}$. Then a rotation matrix $\mathbf{R} \in \mathbb{R}^{k\times k}$ is learned by a Procrustean approach to minimize the quantization error from projected data $\mathbf{Y} = \mathbf{XP}$. The objective function of ITQ can be formulated as follows,

$$\min_{\mathbf{B},\mathbf{R}} \ \|\mathbf{B} - \mathbf{YR}\|_F^2, \ s.t., \ \mathbf{B} = \text{sign}(\mathbf{YR}), \mathbf{RR}^T = \mathbf{I}_k \tag{6}$$

The final projection is given by $\mathbf{PR}$. A rotation matrix satisfies $tr(\mathbf{R}^T\mathbf{YR}) = tr(\mathbf{Y})$ for any square matrix. So $\mathbf{PR}$ must be the optimal solution to problem (3) too. Hence, their basic idea can be summarized as: first minimizing projection error, then minimizing quantization error *while fixing projection error*. Since $\|\mathbf{XP}\|_F^2$ is fixed now, problem (5) in a two-step strategy could be rewritten as formulation below

$$\max_{\mathbf{P}} \ \|\mathbf{XP}\|_1, \ s.t. \ \mathbf{P}^T\mathbf{P} = \mathbf{I}_k, \|\mathbf{XP}\|_F^2 = c \tag{7}$$

where c is the optimal value of problem (3). Here, the extra constraint limits the feasible set of $\mathbf{P}$ hence the obtained projection may be only *sub-optimal* for the second property.

Let us take the toy data in Fig. 2 as an example. Suppose we have two different orthogonal projections $\mathbf{P}_1$ and $\mathbf{P}_2$ for original data $\mathbf{X}$ and map data to 2-dimensional space, and then we solve problem (6) to find the rotation which can minimize the quantization error given the initial projection, whose results are shown in Fig. 2(a) and (b) respectively. The value of $\|\mathbf{XP}_1\|_F^2$ is larger than $\|\mathbf{XP}_2\|_F^2$, indicating that $\mathbf{P}_1$ is a better solution to problem (3) and has smaller projection error based on Eq. (2). However, after an optimal rotation, the quantization error in Fig. 2(a) is *larger* than in Fig. 2(b), implying that the extra constraint from $\|\mathbf{XP}_1\|_F^2 = c$ indeed leads to a worse optimal solution to problem (7). In addition, intuitively we also prefer the result in 2(b). Consequently we can say Fig. 2(b) achieves a better *overall performance* for both properties than Fig. 2(a), even if its projection is indeed not the optimum for the first property.

Based on above observation, it is straightforward to raise a question: *can we sacrifice some projection error to reduce quantization error such that the overall performance is optimized*? We carried out experiment to empirically analyze this question. We utilize a real-world dataset, SIFT1M, which consists of 128-dimensional SIFT points, and we select 10,000 points for our experiment. We compare ITQ that adopts greedy two-step strategy mentioned above, with OPH who optimizes the overall errors in a unified formulation which will be introduced latter. The projection error defined in Eq. (2) and quantization error in Eq. (4) of ITQ and OPH with different Hashcode length are summarized in Table 1. We can observe that OPH slightly increases the projection error (less than 3%) but significantly reduces the quantization error (more than 10% in average). Experiment results introduced in Section 5 also demonstrate that the overall performance of OPH is better. This result suggests that 1) we have a *positive* answer to above question, and in practice, it is *possible* to reduce much quantization error while increase little projection error, i.e., we can find a better trade-off between them such that the overall error is minimized; and 2) the two-step strategy indeed leads to *sub-optimal* solution because the results from initial projection limit the feasible solution set for problem (7). Considering the ultimate goal is to generate binary Hashcodes from original features, the projection learning should take into account both preserving information and minimization quantization error simultaneously, but not separately as ITQ. Therefore it is more reasonable to learn an optimal projection function via *jointly optimizing* the projection and quantization error.

Table 1
e_p and e_q ($\times 10^4$) on SIFT1M.

	e_p			e_q		
	ITQ	OPH	Δ_{OPH}	ITQ	OPH	Δ_{OPH}
16 bits	17.69	17.95	+1.47%	13.97	12.74	−8.80%
32 bits	10.29	10.56	+2.62%	14.31	12.51	−12.58%
64 bits	3.82	3.92	+2.61%	19.22	16.78	−12.70%
96 bits	1.17	1.20	+2.56%	28.07	25.11	−10.47%

3. Learning optimized projection

3.1. Objective function

The observation in Section 2 is based on PCA projection. However, the phenomenon can be observed for other projections, such as in Spectral Hashing [46]. To make our formulation general, i.e., it can be incorporated into different projections, we first need to investigate the projection and quantization error for different projections. Observed from literatures [19,23,34,46,50], the following projection learning formulation is widely adopted,

$$\max_{\mathbf{P}} \ tr(\mathbf{P}^T\mathbf{X}^T\mathbf{WXP}), \ s.t. \ \mathbf{P}^T\mathbf{P} = \mathbf{I}_k \tag{8}$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a weight matrix. Actually, according to the specific situations, we need to preserve different information in original data hence different $\mathbf{W}$ can be adopted. For example, when global Euclidean structure is concerned about [10,19,21], like in PCA, we set $\mathbf{W} = \mathbf{I}_k$ and problem (8) is identical to problem (3); when we want to exploit the local manifold structure, i.e., we want to preserve the *local* neighborhood relationship between data, like Spectral Hashing [46], Anchor Graph Hashing [34] and Locality Preserving Hashing [52], we can adopt the *normalized* adjacency matrix constructed from nearest neighbor graph; to preserve the pairwise label information, such as in Semi-supervised Hashing [45], we can adopt the label-sharing matrix.

With the formulation in (8), the projection error is no longer just the reconstruction error in Euclidean space. But we can define the projection error analogous to PCA below,

Definition 1. Given a weight matrix $\mathbf{W}$, the projection error is defined as the loss of weighted similarity among data,

$$\begin{aligned} e_p &= \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\mathbf{x}_i\mathbf{x}_j^T - \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(\mathbf{x}_i\mathbf{P})(\mathbf{x}_j\mathbf{P})^T \\ &= tr(\mathbf{X}^T\mathbf{WX}) - tr(\mathbf{P}^T\mathbf{X}^T\mathbf{WPX}), \ \mathbf{P}^T\mathbf{P} = \mathbf{I}_k \end{aligned} \tag{9}$$

It is not difficult to verify that the e_p defined in Eq. (2) is a special case of (9) with $\mathbf{W} = \mathbf{I}_n$. Also, since the first term is fixed, minimizing projection error in above definition is equivalent to problem (8). Furthermore, here is another explanation for above definition. The essential purpose of Hashing is to *preserve data similarity*, i.e., the similar points should be similar after projection. Therefore, given a similarity measure (in this paper, we adopt the weighted inner product as [45]), the overall loss of data similarity resulted from a projection reflects how well it preserves the similarity.

The generalized projection error in Definition 1 considers the information preserving property, and the quantization error in Eq. (4) considers the other property. Intuitively, we can jointly optimize them such that the overall result is better, as illustrated in Table 1. Specifically, we can define the overall error under a projection $\mathbf{P}$ as the weighted sum of projection and quantization error as follows

$$e = \lambda e_p + e_q = c + tr(\mathbf{P}^T\mathbf{X}^T(\mathbf{I}_n - \lambda\mathbf{W})\mathbf{X}\mathbf{P}) - 2\|\mathbf{X}\mathbf{P}\|_1 \tag{10}$$

where $c = tr(\mathbf{X}^T\mathbf{W}\mathbf{X}) + nk$ is a constant, and λ is the weight parameter. Therefore we just need to minimize such overall error to learn optimized projection. Based on the unified formulation, we can obtain the objective function for *Optimized Projection* in a general learning framework as

$$\max_{\mathbf{P}} \ \mathcal{O} = tr(\mathbf{P}^T\mathbf{X}^T\mathbf{A}\mathbf{P}\mathbf{X}) + \|\mathbf{X}\mathbf{P}\|_1, \ s.t. \ \mathbf{P}^T\mathbf{P} = \mathbf{I}_k \tag{11}$$

where $\mathbf{A} = \frac{1}{2}(\lambda\mathbf{W} - \mathbf{I}_n)$. As we have mentioned, we formulate this framework to be general such that it can be incorporated with different Hashing methods given specific $\mathbf{W}$. Setting $\mathbf{W} = \mathbf{I}_n$, we obtain the learning problem below

$$\max_{\mathbf{P}} \ \mathcal{O}_{\text{PCA}} = \alpha\|\mathbf{X}\mathbf{P}\|_F^2 + \|\mathbf{X}\mathbf{P}\|_1, \ s.t. \ \mathbf{P}^T\mathbf{P} = \mathbf{I}_k \tag{12}$$

where $\alpha = \frac{1}{2}(\lambda - 1)$. Above is the joint optimization framework of PCA, which optimizes the overall performance of preserving global Euclidean structure and minimizing quantization error, which is intrinsically different from ITQ which optimizes them separately. Our results shown in Table 1 are obtained by it. We can see we do not aim at optimizing either of it, but focus on finding a good *trade-off* which, compared to ITQ, works slightly worse for projection but much better for quantization. Such formulation looks simple but not trivial. To our best knowledge, we are the first to *notice and analyze* the connection between the projection error and quantization error, and *empirically prove* the reasonability and effectiveness of (12).

Another important information in data is the local manifold structure [2]. In CBDR task, we care more about obtaining data sharing the same semantic label as the query where manifold distance is a better measure than global Euclidean distance [34]. Preserving the manifold structure is always formulated as preserving the local nearest neighbor (NN) relationship in data. Specifically, we can construct a p-NN graph with the weight of each edge as below

$$S_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma}}, & \text{if } \mathbf{x}_i \in \mathcal{N}(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where $\mathcal{N}(\mathbf{x}_i)$ is the p-NN of $\mathbf{x}_i$ and σ is the band width [13]. Then we obtain a diagonal degree matrix $\mathbf{D}$ whose diagonal elements are $D_{ii} = \sum_{j=1}^{n} S_{ij}$. Now we can define $\mathbf{W}$ as the *normalized* adjacency matrix for the p-NN graph

$$\mathbf{W} = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2} \tag{14}$$

With this symmetric normalized adjacency matrix, we can learn an optimized projection by (11) which can find a good trade-off between preserving local manifold structure and minimizing quantization error. We can also utilize other definitions of $\mathbf{W}$ for specific purposes. But because of the limitation of space, in this paper we just incorporate OPH with two of the most widely used deifnitions. These two simple definitions can yet lead to state-of-the-art performance.

3.2. Learning algorithm

Considering problem (12) is just a special case of problem (11), we only show the learning algorithm for (11). In this paper, we use the gradient flow [47] to solve the orthogonality constrained ℓ_1 norm regularized maximization problem. Obviously, we can also utilize some accelerated proximal gradient methods (APG) for optimization [25,26]. But to make the algorithm more general, we still use the original one. The basic idea is to firstly find the upgradient at a point, secondly project the upgradient to the tangent space of feasible set defined by the orthogonality constraint, and thirdly move the point with a properly small step size towards this direction *in the feasible set*. We can iterate above three steps until convergence. Specifically, the feasible set for the solution is defined as $\mathcal{M}_{\mathbf{P}} = \{\mathbf{P} \in \mathbb{R}^{d\times k} : \mathbf{P}^T\mathbf{P} = \mathbf{I}_k\}$. Given point $\mathbf{P}_t \in \mathcal{M}_{\mathbf{P}}$, we first compute the *upgradient* of $\mathcal{O}$ at $\mathbf{P}_t$

$$\mathbf{U}_t = -\mathcal{D}\mathcal{O}(\mathbf{P}_t) = -\mathbf{X}^T(\mathbf{A}\mathbf{X}\mathbf{P}_t + \text{sign}(\mathbf{X}\mathbf{P}_t)) \tag{15}$$

To project $\mathbf{U}_t$ to tangent space, we need the theorem below,

Theorem 1. *Given a direction $\mathbf{U}_t$ at $\mathbf{P}_t$, the projection of $\mathbf{U}_t$ onto the tangent space of $\mathcal{M}_{\mathbf{P}}$ at $\mathbf{P}_t$ is computed below*

$$\mathbf{D}_t = \mathbf{M}_t\mathbf{P}_t, \ \text{where } \mathbf{M}_t = \mathbf{U}_t\mathbf{P}_t^T - \mathbf{P}_t\mathbf{U}_t^T \tag{16}$$

The detailed proof can be found in [47]. Then we need to move $\mathbf{P}_t$ to a new point $\mathbf{P}_{t+1}$. Directly moving it like in conventional gradient descent ($\mathbf{P}_{t+1} = \mathbf{P}_t - \tau\mathbf{D}_t$) will move $\mathbf{P}_{t+1}$ out of the feasible set. So in practice [9,44,47], we will compute the next point by the Crank–Nicolson-like scheme:

$$\mathbf{P}_{t+1} = \mathbf{P}_t - \tau\mathbf{M}_t\left(\frac{\mathbf{P}_t + \mathbf{P}_{t+1}}{2}\right) \tag{17}$$

which can lead to the following closed form solution for $\mathbf{P}_{t+1}$

$$\mathbf{P}_{t+1} = (\mathbf{I}_d + \frac{\tau}{2}\mathbf{M}_t)^{-1}(\mathbf{I}_d - \frac{\tau}{2}\mathbf{M}_t)\mathbf{P}_t \tag{18}$$

Above updating rule is called Cayley transformation and τ is a step size satysfying Armijo–Wolfe conditions [36]. Considering that $\mathbf{U}_t$ is a skew-symmetric matrix, i.e., $\mathbf{U}_t^T = -\mathbf{U}_t$, the matrix $\mathbf{I}_d + \frac{\tau}{2}\mathbf{M}_t$ is definitely invertable and $\mathbf{P}_{t+1}$ is also orthogonal, i.e., $\mathbf{P}_{t+1} \in \mathcal{M}_{\mathbf{P}}$, and it results in nonincreasing objective function value. For more detail please see the proof to Lemma 3 in [47]. We can randomly generate an orthogonal matrix to initialize $\mathbf{P}$ and repeat above steps until a stationary point is achieved, i.e., $\mathbf{P}_{t+1} = \mathbf{P}_t$, which is the solution to problem (11). The overall learning algorihm for Optimized Projection is summarized in Algorithm 1. To this end, the

Algorithm 1 Learning optimized projection.

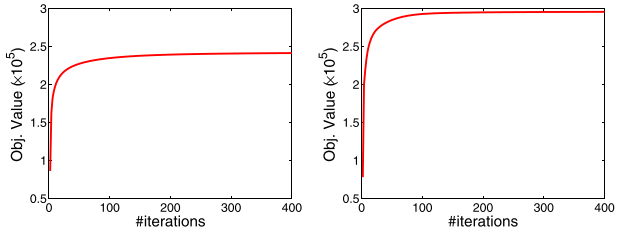
Input:
Training matrix $\mathbf{X}$, Hashcode length k,
weight matrix $\mathbf{W}$, balance parameter λ
Output:
Optimized Projection $\mathbf{P}$

```
Construct A = 1/2(λW − I_n);
Initialize P_0 by a random orthogonal matrix, t=0
repeat
    Compute the upgradient U_t by (15);
    Compute the skew-symmetric matrix M_t by (16);
    Compute the new point P_{t+1} by (18);
    t = t + 1;
until Convergence.
Return P_t;
```

Hashing function is given as $h(\mathbf{x}) = \text{sign}(\mathbf{x}\mathbf{P})$.

4. Discussion

Now we discuss the time complexity of Algorithm 1. The time complexity to compute the upgradient $\mathbf{U}_t$ is $\mathcal{O}(ndk + n^2k)$, to compute the skew-symmetric matrix is $\mathbf{O}(d^2k)$. And to compute new point by Eq. (18), the complexity is $\mathcal{O}(d^2k + dk^2 + k^3)$. Actually, the complexity for inverting an arbitrary $d \times d$ matrix should



(a) CIFAR-10, 64 bits (b) Reuters, 32 bits

Fig. 3. Objective function w.r.t. #iterations.

be $\mathcal{O}(d^3)$. However, since we always have $k \ll d$, especially for high-dimensional image and text data, the rank of matrix $\mathbf{M}_t$ is at most $2k$. Hence following the Sherman–Morrison–Woodbury theorem [40], the complexity for inverting $(\mathbf{I}_d + \frac{\tau}{2}\mathbf{M}_t)$ is $\mathcal{O}(dk^2 + k^3)$. Therefore, the overall complexity for Algorithm 1 is $\mathcal{O}(t(ndk + n^2k + d^2k + dk^2 + k^3))$, where t is the number of iterations to convergence. In Fig. 3, we plot the objective function value w.r.t. the number of iterations on two real-world datasets under different Hashcode length. Here we set $\mathbf{W} = \mathbf{I}_n$, i.e., we incorporate OPH with PCA. We can observe that the objective function value increases steadily with more iterations, and it can converge within 200 iterations, validating the effectiveness of Algorithm 1, which also guarantees the training efficiency.

In above section, we have mentioned that the proposed OPH can be incorporated with different Hashing methods who have the learning formulation in (8) with specific $\mathbf{W}$, such as PCA Hashing [19] and Spectral (Anchor Graph) Hashing [34,46], and etc. Actually, those Hashing methods can be regared as the special cases of OPH with $\lambda \to \infty$, i.e., they only focus on preserving information while ignoring the quantization error. And there are three Hashing methods recently having close relation to OPH. The first is Iterative Quantization (ITQ) [10] which we have introduced in Section 2. It adopts the greedy two-step learning strategy considering two properties separately resulting in sub-optimal solution. The second is Isotropic Hashing (IsoH) [21]. It is also two-step method combining PCA and an extra rotation to balance the variance of each dimension. Actually, balancing the variance can also reduce the quantization error thus we can regard it as a variant of ITQ hence its solution is sub-potimal. In addition, it is unstable in large-scale and high-dimensional data [49]. The third is Harmonious Hashing (HamH) [49] which is derived from Spectral Hashing. It finds a rotation to minimize the distance between the rotated data and a matrix whose variance of each dimension is close. However, such strict requirement and its non-iterative optimization algorithm may fail to find a good enough solution. Above three methods all adopt two-step strategy so their overall performance is worse than OPH. Furthermore, those three methods are derived from specific projection and there is no clue that they can still perform satisfactorily when combined with other projections. But OPH is quite general and based on the generalized projection error defined in Eq. (9). It is robust to different projections, which will be demonstrated in our experiments.

5. Experiment

5.1. Baselines, metrics and settings

We utilize the following related Hashing methods. Locality Sensitive Hashing (LSH) [1], PCA Hashing (PCAH) [19], Spectral Hashing (SpH) [46], Anchor Graph Hashing (AGH) [34] with two-layer Hashing function, Iterative Quantization (ITQ) [10], Isotropic Hashing (IsoH) [21], and Harmonious Hashing (HamH) [49]. For meaningful comparison, we carefully tuned the model parameters for all baselines and the best performance is shown.

Recently, some deep learning based hashing approaches [24,28,29,48,54] have achieved promising results. However, it should be noted that they mostly focus on image hashing and using raw pixels as input, while our approaches and selected baselines focus on feature hashing which is more flexible so that they can use any kinds of features as input. Therefore, we do not choose deep hashing approaches as baselines.

We adopt mean Average Precision (mAP) as the numeric evaluation metric. mAP shows good discriminative power and stability to evaluate the performance of retrieval task. A larger mAP indicates better performance that true positive samples have higher rank. Given a query and R retrieved samples based on Hamming ranking, the Average Precision (AP) is

$$\mathrm{AP} = \frac{1}{L}\sum_{r=1}^{R} P(r)\delta(r) \tag{19}$$

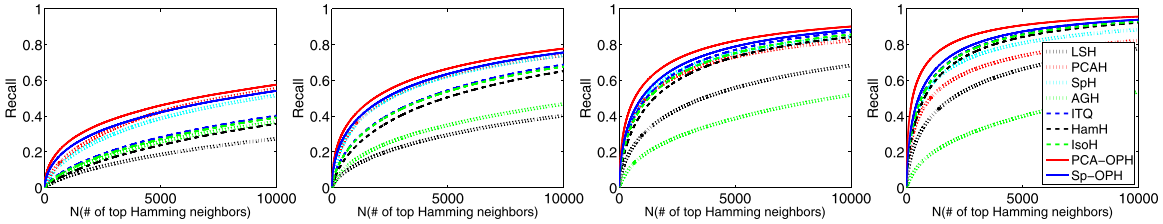
where L is the number of true positive samples in the retrieved set, $P(r)$ denotes the precision of top r retrieved samples defined as the ratio between the number true positive samples and the number of retrieved samples (i.e., r), and $\delta(r)$ is an indicator function which is equal to 1 if the r-th sample is true positive or 0 otherwise. Averaging the AP of all queries leads to mAP. We also adopt the Precision-Recall curve and the Recall curve.

In this paper, we implement two Hashing methods based on OPH. The first utilizes PCA projection, i.e., the projection is learned by problem (12). When implementing it, the parameter α is chosen from {0.01, 0.1, 1} based on the change in projection and quantization error in training data compared to ITQ. We select the value which maximizes the sum of Δ_{OPH} for e_p and e_q. This method is denoted as PCA-OPH. The second considers the local manifold structure, i.e., the weight matrix $\mathbf{W}$ is defined as in Eq. (14) and constructed from a p-NN graph where we set p as 0.1% of training data. The parameter λ is chosen from {0.1, 1, 2, 5, 10}. This method is denoted as Sp-OPH following **Sp**ectral Hashing. When learning $\mathbf{P}$ with Algorithm 1, we stop at the 200th itertation. The binary Hashcodes of a new coming sample $\mathbf{x}$ is computed by $h(\mathbf{x}) = \mathrm{sign}((\mathbf{x} - \bar{\mathbf{x}})\mathbf{P})$, where $\bar{\mathbf{x}}$ is the mean value of training data which is utilized to centralize training data. For fair comparison, our baseline methods also adopt such schema for generating Hashcodes. When compute mAP, we set $R = 50$. To remove any randomness caused by random initialization or random training data selection, all results are the average over 10 repeated runs. All experiments are carried out on a computer which equips Intel Core i7-2600 CPU @3.40 GHz and 16GB RAM.

5.2. Approximate nearest neighbor search

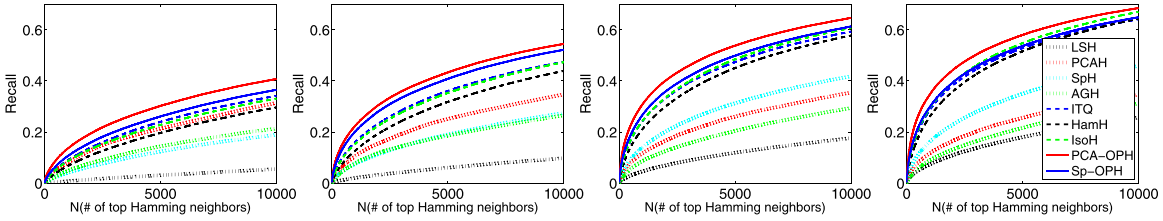
5.2.1. Datasets

The ANN search is a practical and important task in real world. Its purpose is to find some Euclidean neighborhood from database for a given query. In this task, we adopt two widely used large-scale and high-dimensional datasets. The first dataset is SIFT1M [18] which consists of 1 million 128-dimension SIFT [35] points and 10,000 independent points as the query. The second is GIST1M [43] containing 1 million 960-dimension GIST [37] poits and 1000 independent queries. Following [10,21,49], for each query, its true positive samples are the first 100 nearest neighbors in database obtained by brute force search with Euclidean distance. And to test the ability of different Hashing methods to deal with out-of-sample data, we randomly select 10,000 points from database as the training data to learn Hashing functions. Then we generate Hashcodes for samples in both database and query set by the learned Hashing functions as in [5,41].



(a) SIFT1M, 16 bits (b) SIFT1M, 32 bits (c) SIFT1M, 64 bits (d) SIFT1M, 96 bits

Fig. 4. ANN search. Recall curves on SIFT1M with different Hashcode length.



(a) GIST1M, 16 bits (b) GIST1M, 32 bits (c) GIST1M, 64 bits (d) GIST1M, 96 bits

Fig. 5. ANN search. Recall curves on GIST1M with different Hashcode length.

5.2.2. Results

The first important result is shown in Table 1, which we have introduced in Section 2. It demonstrates that we can indeed sacrifice some projection performance for much better quantization to promote overall result by a joint optimization framework instead of the greedy two-step strategy.

More extensive results are given in Figs. 4 and 5. We can observe that PCA-OPH achieve best performance in all experiments and markedly outperform the two-step methods (ITQ and IsoH) in most cases. And Sp-OPH also outperforms baselines in most experiments and it shows superior performance to the two-step HamH in 10 out of 12 experiments. Such results validate the effectiveness of OPH for ANN search. Moreover, we can observe the following points.

Firstly, OPH achieves more improvement over ITQ, IsoH and HamH with shorter Hashcodes. Besides, the improvement is more obvious on GIST1M (960 dimensions) than SIFT1M (128 dimensions). The reason is as below. For shorter Hashcodes, OPH needs to pick fewer directions such that it has more freedom hence it can find a better trade-off from a large number of candidates. However, for two-step methods, its feasible set is limited by the initial projection. Actually, fewer directions lead to more limitation therefore the overall solution is farther from the optimum. In addition, we can even observe that ITQ performs worse than PCAH in some cases, typically with short Hashcodes. This phenomenon also demonstrates that the initial projection will limit the following adjustment step and the two-step strategy leads to sub-optimal solution. With longer Hashcodes, ITQ has more freedom after initial projection so we can observe it significantly outperforms PCAH. But it still suffers from some limitation to some extent such that its overall result is worse than OPH. This result validates again the reasonability of the unified formulation adopted in OPH.

Secondly, OPH and methods considering the quantization error, such as ITQ, perform much better with longer Hashcodes, while methods like PCA may perform worse. Such interesting phenomenon has also been observed by previous researchers [10,45,49]. Intuitively, longer Hashcodes can encode more information thus better results are expected. However, the variance in PCA projected data is quite imbalanced and many dimensions in long Hashcodes contain little information such that they may severely degrade the overall quality of long Hashcodes. Because of the quantization step, the information preserved in the projection may be destroyed. So considering the quantization error is important for projection learning.

5.3. Content-based image retrieval

5.3.1. Datasets

Hashing has been widely utilized in Content-based Image Retrieval (CBIR) [5,10,14,34,53]. Different from ANN, CBIR aims to obtain samples from database which are *semantically related* to the query. In this task, we utilize two celebrated benchmarks. The first is CIFAR-10 [22], which consists of 60,000 images from 10 classes such as airplane and dog. Each image is represented by a 512-dimension GIST descriptor. We select 10,000 images as the query set and the remained images form the database. The second one is NUS-WIDE [4] dataset containing 186,577 images from 10 classes. we adopt the deep features for images which are extracted by the ILSVRC2014 challenge winner GoogLeNet [42] pre-trained on ImageNet. Specifically, we adopt the outputs of the last fully-connected layer as the feature for each image which is a 1024-dimensional vector. In this task, the true positive samples are the ones sharing the same semantic label with the query [34,53]. We randomly select 10,000 samples from the database as the training set.

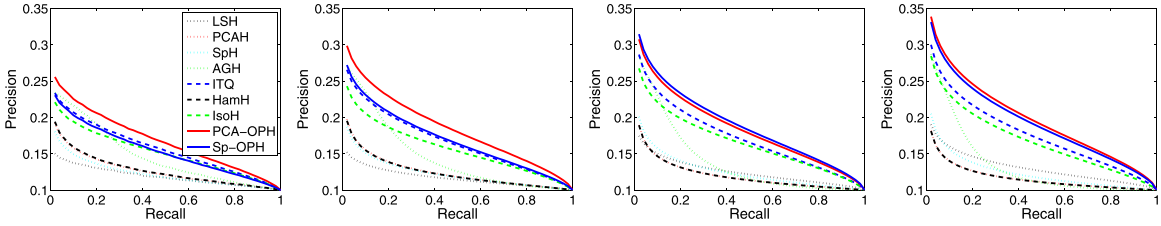
5.3.2. Results

The mAP comparison on two datasets among different methods with various Hashcode length is summarized in Table 2, and the corresponding Precision-recall curves are shown in Figs. 6 and 7. We can observe OPH outperforms other baseline methods regardless of the datasets and Hashcode length, which demonstrates the superiority of OPH for CBIR task. And we have other two important observations.

Firstly, our OPH improves the performance over ITQ and other baselines more with longer Hashcodes. This observation is different from the one in ANN task but in fact they are not contradictory at all. In ANN, the data distribution is simple and we care about the Euclidean neighbors. As we have discussed above, the freedom to select directions is important for ANN. In contrast, CBIR faces more complicated data distribution and we care about high-level semantic relationship between data. In such situation, encoding information effectively becomes the key problem. With short Hashcodes,

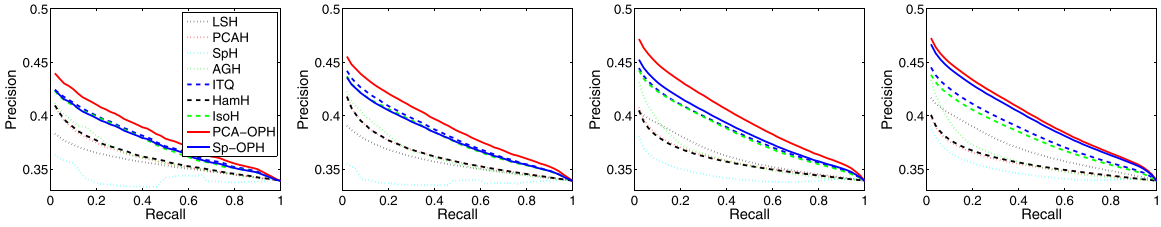
Table 2
mAP comparison. The bold numbers indicate the best two results.

	CIFAR-100						NUS-WIDE					
	16 bits	32 bits	48 bits	64 bits	80 bits	96 bits	16 bits	32 bits	48 bits	64 bits	80 bits	96 bits
LSH	0.2204	0.2372	0.2745	0.2750	0.3002	0.2999	0.4222	0.4397	0.4327	0.4478	0.4575	0.4629
PCAH	0.2962	0.3249	0.3297	0.3340	0.3312	0.3311	0.4853	0.4958	0.4893	0.4865	0.4880	0.4870
SpH	0.2801	0.3064	0.3340	0.3338	0.3495	0.3500	0.4195	0.4220	0.4377	0.4734	0.4569	0.4873
AGH	0.3012	0.3427	0.3487	0.3508	0.3525	0.3524	0.4679	0.4855	0.4881	0.4902	0.4892	0.4912
ITQ	**0.3115**	0.3421	0.3670	0.3708	0.3801	0.3868	0.4832	**0.5083**	0.5032	0.5108	0.5141	0.5132
HamH	0.2963	0.3224	0.3291	0.3486	0.3524	0.3585	**0.4892**	0.4967	0.4892	0.4901	0.4917	0.4863
IsoH	0.3052	0.3397	0.3521	0.3672	0.3759	0.3807	0.4866	0.4966	0.4963	0.4975	0.4973	0.5033
PCA-OPH	**0.3269**	**0.3689**	**0.3863**	**0.3997**	**0.4111**	**0.4222**	**0.4964**	**0.5165**	**0.5283**	**0.5341**	**0.5428**	**0.5486**
Sp-OPH	0.3074	**0.3517**	**0.3769**	**0.4007**	**0.4141**	**0.4185**	0.4871	0.5037	**0.5147**	**0.5190**	**0.5343**	**0.5378**



(a) CIFAR-10, 16 bits (b) CIFAR-10, 32 bits (c) CIFAR-10, 64 bits (d) CIFAR-10, 96 bits

Fig. 6. CBIR. Precision-recall curves on CIFAR-10 with different Hashcode length.



(a) NUS-WIDE, 16 bits (b) NUS-WIDE, 32 bits (c) NUS-WIDE, 64 bits (d) NUS-WIDE, 96 bits

Fig. 7. CBIR. Precision-recall curves on NUS-WIDE with different Hashcode length.

the available information is too little to achieve good performance. But given long Hashcodes, more information is available. Our OPH can preserve more information from original data to Hashcodes because it adopts a unified framework considering the overall performance while the two-step strategy in ITQ, IsoH and HamH can only obtains sub-optimal solution. In addition, the iterative optimization strategy in ITQ and IsoH leads to local optimum. With longer Hashcodes and more variables, their local optimum will be farther from the global optimum therefore the performance gap between OPH and them will be larger.

Secondly, Sp-OPH outperforms PCA-OPH in some cases. Also, SpH and AGH defeat PCAH in several cases. This phenomenon indicates that the local manifold structure is important for CBIR task [34] because samples with short manifold distance tend to have the same semantic label [2]. However, we can observe that the manifold based HamH performs worse than PCA based ITQ and IsoH almost in all cases. Actually, these three methods focus on the adjustment after initial projection. This result implies that the optimization algorithm of HamH is less effective than ITQ and IsoH in practice. We need to say that the adjustment in HamH helps to some extent because HamH is still better than SpH and AGH. This result also reveals the importance of optimization. The superior results of Sp-OPH over ITQ and IsoH show our unified learning framework indeed works.

5.4. Content-based text retrieval

5.4.1. Datasets

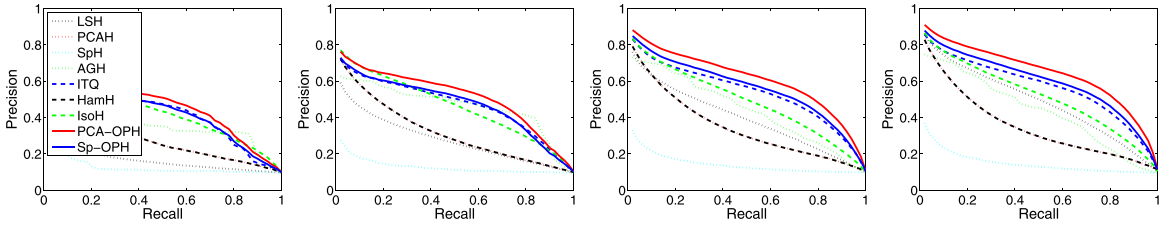
The Content-based Text Retrieval (CBTR) is analogous to CBIR except it is for text data which is another application of Hashing [38]. In this task, two benchmark datasets are involved. The first is TDT2 [17] collected from newswires, radio and television programs. It contains 64,527 documents classified into 100 semantic categories. Each document is represented by a term frequency-inverted document frequency (tf-idf) vector with 36,771 dimensions. The other is Reuters [27] dataset which consists of 21,578 documents from 135 categories. For each document, we use a 18,933-dimension tf-idf vector as original feature. For both datasets, we randomly select 1000 documents as the query set and the remained form the database. We construct the training set by randomly selecting 10,000 documents from database. True positives are documents sharing labels with the query.

5.4.2. Results

In Table 3, we show the mAP of all methods, and the Precision-recall curves are plotted in Figs. 8 and 9. Again, OPH can achieve better performance than baseline methods, especialy PCA-OPH. And some trends similar to CBIR appear in CBTR too. For example, OPH achieves more improvement given longer Hashcodes. Actually, this phenomenon is more reasonable in CBTR because text data has higher dimensionality such that we need longer codes to capture the information. However, Sp-OPH is consistently worse

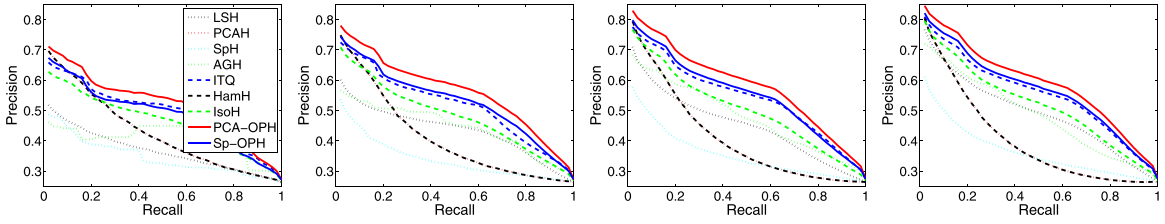
Table 3
mAP comparison. The bold numbers indicate the best two results.

	TDT2						Reuters					
	16 bits	32 bits	48 bits	64 bits	80 bits	96 bits	16 bits	32 bits	48 bits	64 bits	80 bits	96 bits
LSH	0.4270	0.6368	0.7447	0.7750	0.8246	0.8515	0.5918	0.6439	0.6902	0.7298	0.7448	0.7717
PCAH	0.6488	0.7348	0.7455	0.7672	0.8047	0.8105	0.6844	0.7203	0.7579	0.7978	0.7964	0.7990
SpH	0.3291	0.3678	0.4103	0.4726	0.4783	0.5082	0.4977	0.6162	0.6415	0.6592	0.6641	0.6739
AGH	0.4183	0.6181	0.7416	0.7488	0.7659	0.7683	0.4513	0.5821	0.6569	0.7150	0.7282	0.7297
ITQ	0.6629	0.7529	0.7902	0.8368	0.8286	0.8638	0.6977	**0.7724**	**0.7916**	0.8012	0.8061	0.8172
HamH	0.6788	0.7548	0.7855	0.7972	0.8147	0.8205	**0.7344**	0.7703	0.7879	0.7978	0.7964	0.7990
IsoH	**0.7032**	**0.7683**	0.8131	0.8349	0.8523	0.8604	0.6826	0.7518	0.7739	0.7849	0.7992	0.8042
PCA-OPH	**0.7288**	**0.7740**	**0.8428**	**0.8727**	**0.8978**	**0.9094**	**0.7357**	**0.7830**	**0.8162**	**0.8376**	**0.8489**	**0.8507**
Sp-OPH	0.6807	0.7641	**0.8205**	**0.8495**	**0.8602**	**0.8785**	0.6537	0.7682	0.7859	**0.8144**	**0.8231**	**0.8375**



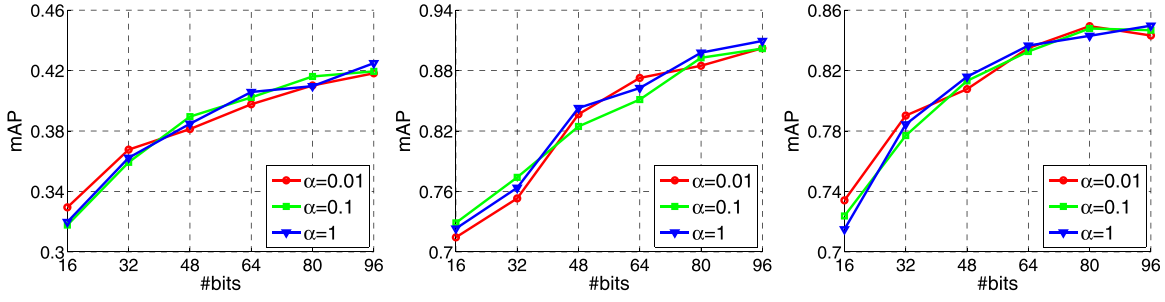
(a) TDT2, 16 bits (b) TDT2, 32 bits (c) TDT2, 64 bits (d) TDT2, 96 bits

Fig. 8. CBTR. Precision-recall curves on TDT2 with different Hashcode length.



(a) Reuters, 16 bits (b) Reuters, 32 bits (c) Reuters, 64 bits (d) Reuters, 96 bits

Fig. 9. CBTR. Precision-recall curves on Reuters with different Hashcode length.



(a) CIFAR-10 (b) TDT2 (c) Reuters

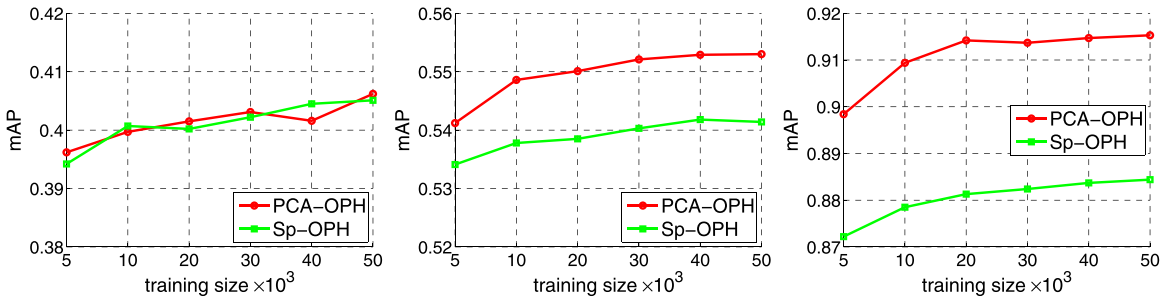
Fig. 10. mAP of PCA-OPH w.r.t. α.

than PCA-OPH in all experiments for CBTR. One possible reason might be that the p-NN graph is less precise in high-dimensional data like text given the imprecise distance computing, and the local manifold structure can not be effectively exploited with such low-quality graph. But in PCA-OPH, we do not need to compute the distance *between* data thus such problem can be avoided.

5.5. Other issues

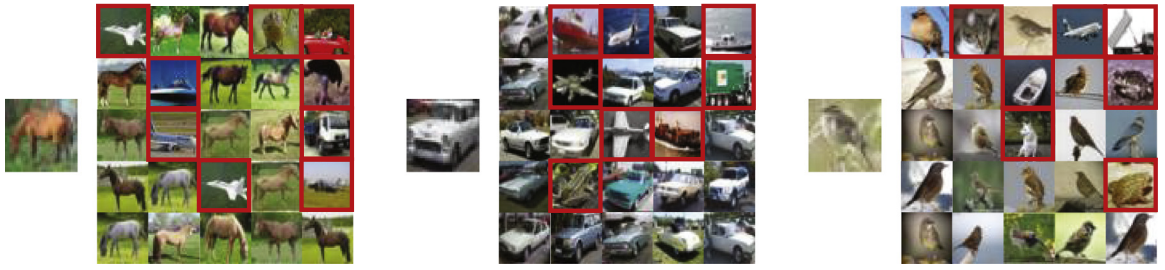
Now we investigate the balance parameter in OPH. For simplicity, we show the effect of α on PCA-OPH. This parameter controls the balance of projection error and quantization error. Results on CIFAR-10, TDT2 and Reuters with different α are plotted in Fig. 10. Indeed, the value of α has influence on OPH to some extent, but we can also notice that statisfactory trade-off can be obtained with a proper α. Typically, $\alpha \in [0.01, 1]$ always works. Actually, if α is too big (like 10,000), PCA-OPH degenerates to PCAH.

Then we show the effect of the size of training data. The performance of PCA-OPH and Sp-OPH on CIFAR-10, NUS-WIDE and TDT2 is plotted in Fig. 11. Given more training data, OPH can perform slightly better because more information is available. But it is also obvious that the improvement is quite small when the size is larger than 10,000. When the training size grows from 10,000 to 50,000, the improvement in mAP is less than 0.01. In fact, given



(a) CIFAR-10, 64 bits (b) NUS-WIDE, 96 bits (c) TDT2, 96 bits

Fig. 11. mAP of OPH w.r.t. training size.



(a) 32 bits (b) 64 bits (c) 96 bits

Fig. 12. Top 5 retrieved images for query (left) by LSH (up), PCAH, AGH, ITQ, and PCA-OPH (down).

enough training data (say, 10,000), the model can be well trained and extra data is redundant. This phenomenon is also observed in baseline methods, like PCAH, ITQ, AGH, and etc.

6. Conclusion

Previous projection learning methods for Hashing consider preserving information and minimizing quantization error seperately. In this paper, we empirically study this problem and prove that only sub-optimal solution can be achieved in this two-step strategy. Hence we propose a unified and general projection learning framework for Hashing to find the best trade-off between them and learn an Optimized Projection for better overall performance. An effective optimization algorithm is given. Extensive experiments for ANN and CBDR on several benchmarks compared to state-of-the-art related Hashing methods validate the effectiveness of OPH (Fig. 12).

Acknowledgments

This research was supported by the National Natural Science Foundation of China grant no. 61571269.

References

[1] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in: FOCS'06, 2006, pp. 459–468.
[2] D. Cai, X. He, J. Han, T.S. Huang, Graph regularized nonnegative matrix factorization for data representation, IEEE TPAMI 33 (8) (2011) 1548–1560.
[3] L. Chen, D. Xu, I.W. Tsang, X. Li, Spectral embedded hashing for scalable image retrieval, IEEE T. Cybern. 44 (7) (2014) 1180–1190.
[4] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, Y. Zheng, NUS-WIDE: a real-world web image database from national university of Singapore, CIVR, 2009.
[5] G. Ding, Y. Guo, J. Zhou, Collective matrix factorization hashing for multimodal data, in: CVPR, 2014, pp. 2083–2090.
[6] G. Ding, Y. Guo, J. Zhou, Y. Gao, Large-scale cross-modality search via collective matrix factorization hashing, IEEE Trans. Image Process. 25 (11) (2016) 5427–5440.
[7] J.H. Friedman, J.L. Bentley, R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Math. Software 3 (1977) 209–226.
[8] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: VLDB, 1999, pp. 518–529.
[9] D. Goldfarb, Z. Wen, W. Yin, A curvilinear search method for the p-harmonic flow on spheres, SIAM J. Imaging Sci. 2 (1) (2009) 84–109.
[10] Y. Gong, S. Lazebnik, Iterative quantization: a procrustean approach to learning binary codes, in: CVPR, 2011, pp. 817–824.
[11] Y. Guo, G. Ding, J. Han, Robust quantization for general similarity search, IEEE Trans. Image Processing 27 (2) (2018) 949–963.
[12] Y. Guo, G. Ding, J. Han, Y. Gao, Sitnet: Discrete similarity transfer network for zero-shot hashing, in: IJCAI, 2017, pp. 1767–1773.
[13] Y. Guo, G. Ding, J. Han, Y. Gao, Zero-shot learning with transferred samples, IEEE Trans. Image Process. 26 (7) (2017) 3277–3290.
[14] Y. Guo, G. Ding, X. Jin, J. Wang, Learning predictable and discriminative attributes for visual recognition, in: AAAI, 2015, pp. 3783–3789.
[15] Y. Guo, G. Ding, L. Liu, J. Han, L. Shao, Learning to hash with optimized anchor embedding for scalable retrieval, IEEE Trans. Image Process. 26 (3) (2017) 1344–1354.
[16] K. He, F. Wen, J. Sun, K-means hashing: an affinity-preserving quantization method for learning binary compact codes, in: CVPR, 2013, pp. 2938–2945.
[17] J-Fiscus, G.R. Doddington, J.S. Garofolo, A.F. Martin, Nist's 1998 topic detection and tracking evaluation (TDT2), EUROSPEECH, 1999.
[18] H. Jegou, M. Douze, C.S. and, Product quantization for nearest neighbor search, TPAMI 33 (1) (2011) 117–128.
[19] H. Jegou, M. Douze, C. Schmid, P. Pérez, Aggregating local descriptors into a compact image representation, in: CVPR, 2010, pp. 3304–3311.
[20] W. Kong, W.-J. Li, Double-bit quantization for hashing., AAAI, 2012.
[21] W. Kong, W.-J. Li, Isotropic hashing, in: NIPS, 2012, pp. 1655–1663.
[22] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Tech Report, Univ. of Toronto, 2009.
[23] S. Kumar, R. Udupa, Learning hash functions for cross-view similarity search, in: IJCAI, 2011, pp. 1360–1365.
[24] H. Lai, Y. Pan, Y. Liu, S. Yan, Simultaneous feature learning and hash coding with deep neural networks, in: CVPR, 2015, pp. 3270–3278.
[25] X. Lan, A.J. Ma, P.C. Yuen, R. Chellappa, Joint sparse representation and robust feature-level fusion for multi-cue visual tracking, IEEE Trans. Image Process. 24 (12) (2015) 5826–5841.
[26] X. Lan, S. Zhang, P.C. Yuen, R. Chellappa, Learning common and feature-specific patterns: a novel multiple-sparse-representation-based tracker, IEEE Trans. Image Process. 27 (4) (2018) 2022–2037.
[27] D.D. Lewis, Y. Yang, T.G. Rose, F. Li, RCV1: A new benchmark collection for text categorization research, JMLR (2004).
[28] V.E. Liong, J. Lu, G. Wang, P. Moulin, J. Zhou, Deep hashing for compact binary codes learning, in: CVPR, 2015, pp. 2475–2483.
[29] H. Liu, R. Wang, S. Shan, X. Chen, Deep supervised hashing for fast image retrieval, CVPR, 2016.

[30] L. Liu, L. Shao, F. Shen, M. Yu, Discretely coding semantic rank orders for supervised image hashing, in: CVPR, 2017, pp. 5140–5149.
[31] L. Liu, F. Shen, Y. Shen, X. Liu, L. Shao, Deep sketch hashing: Fast free-hand sketch-based image retrieval, in: CVPR, 2017, pp. 2298–2307.
[32] L. Liu, M. Yu, L. Shao, Projection bank: From high-dimensional data to medium-length binary codes, in: ICCV, 2015, pp. 2821–2829.
[33] L. Liu, M. Yu, L. Shao, Latent structure preserving hashing, Int. J. Comput. Vis. 122 (3) (2017) 439–457.
[34] W. Liu, J. Wang, S. Kumar, S.F. Chang, Hashing with graphs, in: ICML, 2011, pp. 1–8.
[35] D.G. Lowe, Distinctive image features from scale-invariant keypoints, IJCV 60 (2) (2004) 91–110.
[36] J. Nocedal, S. Wright, Numerical Optimization, 1999.
[37] A. Oliva, T. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, IJCV 42 (2001) 145–175.
[38] R. Salakhutdinov, G. Hinton, Semantic hashing, IJAR 50 (7) (2009) 969–978.
[39] F. Shen, Y. Xu, L. Liu, Y. Yang, Z.H.H.T. Shen, Unsupervised deep hashing with similarity-adaptive and discrete optimization, IEEE TPAMI (2018).
[40] J. Sherman, W. Morrison, Adjustment of an inverse matrix corresponding to a change in one element of a given matrix, Ann. Math. Stat. (1950).
[41] J. Song, Y. Yang, Y. Yang, Z. Huang, H.T. Shen, Inter-media hashing for large-scale retrieval from heterogeneous data sources, in: ICMD, ACM, 2013, pp. 785–796.
[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S.E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: CVPR, 2015, pp. 1–9.
[43] A.B. Torralba, R. Fergus, W.T. Freeman, 80 million tiny images: a large data set for nonparametric object and scene recognition, TPAMI 30 (11) (2008) 1958–1970.
[44] L. Vese, S. Osher, Numerical methods for p-harmonic flows and applications to image processing, SIAM J. Numer. Anal. (2002).
[45] J. Wang, S. Kumar, S.-F. Chang, Semi-supervised hashing for scalable image retrieval, in: CVPR, 2010, pp. 3424–3431.
[46] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: NIPS, 2008, pp. 1753–1760.
[47] Z. Wen, W. Yin, A feasible method for optimization with orthogonality constraints, Math. Prog. (2013) 397–434.
[48] R. Xia, Y. Pan, H. Lai, C. Liu, S. Yan, Supervised hashing for image retrieval via image representation learning, in: AAAI, 2014, pp. 2156–2162.
[49] B. Xu, J. Bu, Y. Lin, C. Chen, X. He, D. Cai, Harmonious hashing, IJCAI, 2013.
[50] D. Zhang, F. Wang, L. Si, Composite hashing with multiple information sources, in: SIGIR, 2011, pp. 225–234.
[51] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: SIGIR, 2010, pp. 18–25.
[52] K. Zhao, H. Lu, J. Mei, Locality preserving hashing, in: AAAI, 2014, pp. 2874–2881.
[53] J. Zhou, G. Ding, Y. Guo, Latent semantic sparse hashing for cross-modal similarity search, in: SIGIR, 2014, pp. 415–424.
[54] B. Zhuang, G. Lin, C. Shen, I. Reid, Fast training of triplet-based deep binary embedding networks, CVPR, 2016.