Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Asymmetric Discrete Graph Hashing

Xiaoshuang Shi, Fuyong Xing,
Kaidi Xu, Manish Sapkota, Lin Yang
University of Florida, Gainesville, FL, 32611, USA
xsshi2015@ufl.edu

Abstract

Recently, many graph based hashing methods have been emerged to tackle large-scale problems. However, there exists two major bottlenecks: (1) directly learning discrete hashing codes is an NP-hard optimization problem; (2) the complexity of both storage and computational time to build a graph with n data points is $\mathcal{O}(n^2)$. To address these two problems, in this paper, we propose a novel yet simple supervised graph based hashing method, asymmetric discrete graph hashing, by preserving the asymmetric discrete constraint and building an asymmetric affinity matrix to learn compact binary codes. Specifically, we utilize two different instead of identical discrete matrices to better preserve the similarity of the graph with short binary codes.We generate the asymmetric affinity matrix using m ($m << n$) selected anchors to approximate the similarity among all training data so that computational time and storage requirement can be significantly improved. In addition, the proposed method jointly learns discrete binary codes and a low-dimensional projection matrix to further improve the retrieval accuracy. Extensive experiments on three benchmark large-scale databases demonstrate its superior performance over the recent state of the arts with lower training time costs.

Introduction

Hashing techniques have attracted considerable attention for tackling a variety of large-scale tasks in computer vision and machine learning. The core idea of hashing is to map high-dimensional data into a low-dimensional binary (Hamming) space with preserving the similarity among neighbors (Wang, Kumar, and Chang 2012). Hence the high-dimensional data can be represented by a set of compact binary codes, which lead to *significant gains in both storage and query speed* (Liu et al. 2014).

Over the past decades, various hashing methods (Indyk and Motwani 1998), (Weiss, Torralba, and Fergus 2009), (Liu et al. 2011), (Gong et al. 2013), (Strecha et al. 2012), (Norouzi and Blei 2011), (Zhang et al. 2010) have been proposed. Generally, these methods can be grouped into two main categories: *data-independent* and *data-dependent*. Locality-sensitive hashing (LSH) (Indyk and Motwani 1998) is one of representative data-independent methods, which utilizes simple random projections to generate binary codes, thereby requiring long binary codes to achieve good retrieval accuracy. To learn compact binary codes, an increasing number of methods focus on learning data-dependent hashing functions with available training data. Based on whether utilizing the semantic (label) information, data-dependent methods can be further classified into unsupervised and supervised hashing. Unsupervised hashing methods, such as spectral hashing (SH) (Weiss, Torralba, and Fergus 2009), anchor graph hashing (AGH) (Liu et al. 2011) and iterative quantization (ITQ) (Gong et al. 2013), try to explore intrinsic features of data to learn binary codes without any supervision. By contrast, supervised hashing methods, including linear discriminant analysis hashing (LDAH) (Strecha et al. 2012), minimal loss hashing (MLH) (Norouzi and Blei 2011) and semi-supervised hashing (SSH) (Wang, Kumar, and Chang 2012), leverage semantic information to map the high-dimensional data into a low-dimensional binary space. Recently, supervised hashing has attracted increasing interest, since unsupervised hashing methods are not able to guarantee desired retrieval accuracy via semantic distances due to the semantic gap (Wang, Kumar, and Chang 2012). Meanwhile, compared to linear hashing methods, kernel-based hashing methods like binary reconstruction embedding (BRE) (Kulis and Darrell 2009), random maximum margin hashing (RMMH) (Joly and Buisson 2011), kernel supervised hashing (KSH) (Liu et al. 2012) and kernel supervised discrete hashing (KSDH) (Shi et al. 2016b) usually achieve better retrieval performance because they can capture the non-linear manifold structure hidden in the data.

Most of data-dependent hashing methods are derived from traditional dimensionality reduction techniques. One popular dimensionality reduction framework is graph embedding (Yan et al. 2007), on which many hashing methods like SH and AGH have been developed. However, there are two major bottlenecks for graph based hashing methods: (1) directly learning the discrete binary codes is an NP-hard problem; (2) the complexity of both storage and computational time to build a graph with n data points is $\mathcal{O}(n^2)$. To obtain binary codes, most of graph based hashing methods utilize the *relaxation+rounding* schemes, which might generate accumulate quantization errors between hashing and projection functions, especially for large-scale training data

(Shen et al. 2015). Therefore, discrete graph hashing (DGH) (Liu et al. 2014) with preserving *symmetric discrete constraint* (using the element-wise product of two identical discrete matrices to preserve the similarity among neighbors) has been proposed to remove this limitation. To improve storage requirement and computational time of graph generation, DGH adopts one popular strategy, namely *anchor graphs*, to build an approximate symmetric neighbor graph (Liu et al. 2011). Furthermore, recent literature (Neyshabur et al. 2013) argues that the symmetric discrete constraint might be difficult to optimize and not necessary to learn binary codes of training data, and the *asymmetric discrete constraint* (using the element-wise product of two different discrete matrices to approximate the similar matrix) might be more powerful to learn shorter binary codes. However, the optimization problem in (Neyshabur et al. 2013) is not completely same to the problem in graph based hashing algorithms. In addition, the complexity of both storage and computational time of the algorithm (Neyshabur et al. 2013) is $\mathcal{O}(n^2)$, which requires large computer memory and consumes high training costs for a large n.

Motivated by the aforementioned observations, in this paper, we propose a novel supervised graph based hashing method, asymmetric discrete graph hashing (ADGH), that can utilize the semantic information to encode high-dimensional data into compact binary codes with low training time costs. Specifically, instead of preserving the symmetric discrete constraint, we preserve the *asymmetric discrete constraint* in the proposed optimization model. To reduce the size of the graph, unlike the anchor graphs strategy generating an $n \times n$ symmetric affinity matrix, we utilize a small number m ($m << n$) anchors to build an $n \times m$ *asymmetric affinity matrix* to approximate the similarity among all training data. Besides directly learning binary codes of training data, the proposed method can simultaneously learn the low-dimensional projection matrix. In summary, our contributions are:

- We propose a novel supervised graph based hashing model by preserving the asymmetric discrete constraint and building an asymmetric affinity matrix. In addition, our model can jointly learn binary codes of training data and a low-dimensional projection matrix, which can further improve the retrieval accuracy.
- We present an optimization procedure to solve the proposed model in an alternative and efficient manner, and analyze its convergence and time complexity.
- We evaluate the proposed method on three popular large-scale image and video databases, and demonstrate its superior performance over the state of the arts with lower training time costs.

Graph based asymmetric discrete hashing

Given data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \in \mathbb{R}^{d \times n}$, where d and n are the number of predictors and observations, respectively. Without loss of generality, in this paper, the data $\{\mathbf{x}_i\}_{i=1}^n$ are assumed to be zero centered, i.e. $\sum_{i=1}^n \mathbf{x}_i = 0$. These data points are used to construct a graph $\mathcal{G} = \{\mathbf{X}, \mathbf{W}\}$, where $\mathbf{W}$ is a symmetric affinity matrix with w_{ij} characterizing the favorite similarity relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$. The purpose of traditional graph methods is to map the high-dimensional data into a low-dimensional space with similarity among neighbors best preserved. Suppose that $\mathbf{Y} \in \mathbb{R}^{n \times r}$ is the corresponding low-dimensional data of $\mathbf{X}$, one popular formulation can be written as follows:

$$\min_{\mathbf{Y}} Tr\left\{\mathbf{Y}^T\mathbf{L}\mathbf{Y}\right\}, s.t.\ \mathbf{Y}^T\mathbf{Y} = \mathbf{I}_r, \quad (1)$$

where $\mathbf{L} = \mathbf{S} - \mathbf{W}$ is the graph Laplacian, and $\mathbf{S}$ is a diagonal matrix with the i-th diagonal element $s_{ii} = \sum_j w_{ij}$. Many graph based hashing algorithms (Weiss, Torralba, and Fergus 2009), (Liu et al. 2011) have been developed based on Eq. (1). However, most of them use the relaxation+rounding schemes to learn binary codes, which might generate the accumulated quantization errors between hashing and projection functions. Although DGH directly learn the binary codes via preserving the symmetric discrete constraint, the optimization procedure is relatively difficult to optimize and requires high training time cost. Meanwhile, the symmetric discrete constraint might require longer binary codes than the asymmetric discrete constraint to preserve the similarity of the affinity matrix. For clear illustration, we present a lemma as follows:

Lemma 1: *Given a symmetric positive semidefinite matrix* $\mathbf{M} \in \mathbb{R}^{n \times n}$*, for any discrete matrix* $\mathbf{B}_1 \in \{\pm 1\}^{n \times r}$*, there exists a discrete matrix* $\mathbf{B}_2 \in \{\pm 1\}^{n \times r}$ *such that* $Tr\left\{\mathbf{B}_1^T\mathbf{M}\mathbf{B}_1\right\} \leq Tr\left\{\mathbf{B}_1^T\mathbf{M}\mathbf{B}_2\right\}$. (Proof in the supplemental material)

In Lemma 1, a larger objective value of $Tr\left\{\mathbf{B}_1^T\mathbf{M}\mathbf{B}_2\right\}$ suggests that the discrete matrices can preserve more similarity (label) information, which usually means better retrieval accuracy.

Dimensionality reduction methods like principal component analysis (PCA) (Wright et al. 2009), (Candès et al. 2011) and nonnegative matrix factorizations (Recht et al. 2012) (Zhou, Bian, and Tao 2013), suggest that a matrix with high dimensionality yet low-rank can be represented or approximated by a linear combination of basis vectors. Therefore, in our problem, the affinity matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ can be approximated by a factorization $\mathbf{W} \approx \mathbf{V}\mathbf{P}$, where $\mathbf{V} \in \mathbb{R}^{n \times m}$, $\mathbf{V} \subset \mathbf{W}$ and $\mathbf{P} \in \mathbb{R}^{m \times n}$ is a weight matrix. Typically, $m << n$. Based on these observations, we propose a novel graph based hashing method, namely asymmetric discrete graph hashing (ADGH).

Model formulation

Suppose there exists a projection matrix $\mathbf{A} \in \mathbb{R}^{d \times r}$ such that $\mathbf{X}^T\mathbf{A} = \mathbf{Y}$. Because $\mathbf{Y}^T\mathbf{Y} = \mathbf{I}_r$ and $\mathbf{W} \approx \mathbf{V}\mathbf{P}$, the optimization problem in Eq. (1) can be reformulated as:

$$\max_{\mathbf{A}} Tr\left\{\mathbf{A}^T\mathbf{X}\mathbf{V}\mathbf{P}\mathbf{X}^T\mathbf{A}\right\}, \ s.t.\ \mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A} = \mathbf{I}_r. \quad (2)$$

Although $\mathbf{V}$ can be a subset of $\mathbf{W}$, it is computationally expensive to calculate the matrix $\mathbf{P}$. To avoid the calculation of $\mathbf{P}$, we assume $\mathbf{Z} = \mathbf{P}\mathbf{X}^T \in \mathbb{R}^{m \times d}$ to be anchors. Then

the matrix $\mathbf{V}$ can be viewed as the affinity matrix characterizing the relationship between the data $\mathbf{X}$ and the anchors $\mathbf{Z}$, and the projection matrix $\mathbf{A}$ is to maintain their similarity into a low-dimensional space. To learn compact binary codes of $\mathbf{X}$ and $\mathbf{Z}$ with similarity preserved, we consider the requirement of the maximum information of each hash bit and the minimum redundancy among different hash bits, and formulate the following optimization problem:

$$\begin{aligned} &\max_{\mathbf{B},\mathbf{D}} Tr\left\{\mathbf{B}^T\mathbf{V}\mathbf{D}\right\}, \\ s.t.\ &\mathbf{B}\in\{-1,1\}^{n\times r}, \mathbf{D}\in\{-1,1\}^{m\times r}, \\ &\mathbf{B}^T\mathbf{1}_n=\mathbf{0}, \mathbf{B}^T\mathbf{B}=n\mathbf{I}_r, \\ &\mathbf{D}^T\mathbf{1}_m=\mathbf{0}, \mathbf{D}^T\mathbf{D}=m\mathbf{I}_r, \end{aligned} \tag{3}$$

where $\mathbf{1}_m\in\mathbb{R}^m$ is a column vector with all elements being one, the discrete matrices $\mathbf{B}$ and $\mathbf{D}$ represent the binary codes of the data $\mathbf{X}$ and the anchors $\mathbf{Z}$, respectively.

It is difficult to solve Eq. (3) due to the hard constraints. To compute the binary code matrices $\mathbf{B}$, $\mathbf{D}$ and the projection matrix $\mathbf{A}$, we soften the hard constraints to obtain:

$$\begin{aligned} &\max_{\mathbf{B},\mathbf{D},\mathbf{A}} Tr\left\{\mathbf{B}^T\mathbf{V}\mathbf{D}\right\}-\tfrac{\gamma}{2}\left\|\mathbf{B}-\mathbf{X}^T\mathbf{A}\right\|_F^2, \\ &s.t.\ \mathbf{B}\in\{-1,1\}^{n\times r}, \mathbf{D}\in\{-1,1\}^{m\times r}, \\ &\mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A}=n\mathbf{I}_r, \end{aligned} \tag{4}$$

where the parameter γ is to weight the similarity information and the error between the discrete binary code matrix $\mathbf{B}$ and the projection function $f(\mathbf{X})=\mathbf{X}^T\mathbf{A}$. Since $\mathbf{A}^T\mathbf{X}\mathbf{1}_n=\mathbf{0}$, $\mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A}=n\mathbf{I}_r$ that can reduce the redundancy among data points (Shi et al. 2016a), when $\gamma\to\infty$, the binary matrix $\mathbf{B}$ satisfies the constraint $\mathbf{B}^T\mathbf{1}_n=\mathbf{0}$ and $\mathbf{B}^T\mathbf{B}=n\mathbf{I}_r$. Note that we ignore the influence of the discrete matrix $\mathbf{D}$ on the projection matrix $\mathbf{A}$ due to $m<<n$, hence *it is unnecessary to compute matrix* $\mathbf{P}$. In practice, we can select m data points $\widetilde{\mathbf{X}}\in\mathbb{R}^{d\times m}$ from $\mathbf{X}$ and build $\mathbf{V}$ based on the similarity between the selected data and the training data.

To improve the robustness of the projection function to outliers, we replace the constraint $\mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A}=n\mathbf{I}_r$ with a more general constraint $\left\|\mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A}-n\mathbf{I}_r\right\|_F^2\le\epsilon$, where ϵ is a very small coefficient. Since $Tr\left\{\mathbf{B}^T\mathbf{B}\right\}=rn$ and $Tr\left\{\mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A}\right\}\to rn$, we propose our graph based hashing framework:

$$\begin{aligned} &\max_{\mathbf{B},\mathbf{D},\mathbf{A}} Tr\left\{\mathbf{B}^T\mathbf{V}\mathbf{D}\right\}+\gamma Tr\left\{\mathbf{B}^T\mathbf{X}^T\mathbf{A}\right\}, \\ &s.t.\ \mathbf{B}\in\{-1,1\}^{n\times r}, \mathbf{D}\in\{-1,1\}^{m\times r}, \\ &\left\|\mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A}-n\mathbf{I}_r\right\|_F^2\le\epsilon. \end{aligned} \tag{5}$$

The framework incorporates the binary codes $\mathbf{B}$, $\mathbf{D}$ and the projection matrix $\mathbf{A}$ so that embedding learning and regression model can be jointly implemented and optimized, which can further improve the robustness of the projection matrix $\mathbf{A}$ (Shi et al. 2015), (Hou et al. 2014).

Out-of-Sample Hashing: After obtaining $\mathbf{A}$, given a query data point $\mathbf{q}\in\mathbb{R}^d$, we will use the hashing functions $H(\mathbf{q})=[h_1(\mathbf{q}),h_2(\mathbf{q}),\cdots,h_r(\mathbf{q})]=\left\{sgn(\mathbf{a}_i^T\mathbf{q})\right\}_{i=1}^r$ to achieve its binary codes, where

$$sgn(\mathbf{a}_i^T\mathbf{q})=\begin{cases}1 & if\ \mathbf{a}_i^T\mathbf{q}\ge 0;\\ -1 & otherwise.\end{cases} \tag{6}$$

Optimization process

Since it is difficult to directly to solve the problem in Eq. (5), we propose a tractable optimization manner via alternative solving the following four subproblems.

B-subproblem:

$$\begin{aligned} &\max_{\mathbf{B}} Tr\left\{\mathbf{B}^T\mathbf{V}\mathbf{D}\right\}+\gamma Tr\left\{\mathbf{B}^T\mathbf{X}^T\mathbf{A}\right\}, \\ &s.t.\ \mathbf{B}\in\{-1,1\}^{n\times r}, \end{aligned} \tag{7}$$

from which we can obtain $\mathbf{B}=sgn(\mathbf{V}\mathbf{D}+\gamma\mathbf{X}^T\mathbf{A})$.

D-subproblem:

$$\max_{\mathbf{D}} Tr\left\{\mathbf{B}^T\mathbf{V}\mathbf{D}\right\},\ \ s.t.\ \mathbf{D}\in\{-1,1\}^{m\times r}, \tag{8}$$

whose optimal solution is $\mathbf{D}=sgn(\mathbf{V}^T\mathbf{B})$.

C-subproblem:

$$\max_{\mathbf{C}} Tr\left\{\mathbf{B}^T\mathbf{C}\right\},\ \ s.t.\ \mathbf{C}^T\mathbf{1}_n=\mathbf{0}, \mathbf{C}^T\mathbf{C}=n\mathbf{I}_r, \tag{9}$$

where $\mathbf{C}\in\mathbb{R}^{n\times r}$ is an auxiliary matrix to replace $\mathbf{X}^T\mathbf{A}$. Eq. (9) can be solved based on Theorem 1.

Theorem 1: *If* $rank(\mathbf{J}\mathbf{B})=r$, *where* $\mathbf{J}=\mathbf{1}_n\mathbf{1}_n^T-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$. *Eq. (9) has a unique global solution* $\mathbf{C}=\sqrt{n}\mathbf{R}\mathbf{Q}^T$, $\mathbf{Q}\in\mathbb{R}^{r\times r}$ *is obtained based on the SVD of* $\mathbf{B}^T\mathbf{J}\mathbf{B}=\mathbf{Q}\mathbf{\Sigma}^2\mathbf{Q}^T$ *and* $\mathbf{R}=\mathbf{J}\mathbf{B}\mathbf{Q}\mathbf{\Sigma}^{-1}\in\mathbb{R}^{n\times r}$. (Proof in the supplemental material)

A-subproblem:

$$\min_{A}\left\|\mathbf{C}-\mathbf{X}^T\mathbf{A}\right\|_F^2+\lambda\left\|\mathbf{A}\right\|_F^2, \tag{10}$$

whose solution is $\mathbf{A}=(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I}_d)^{-1}\mathbf{X}\mathbf{C}$. Obviously, $\mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A}=\mathbf{C}^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I}_d)^{-1}\mathbf{X}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I}_d)^{-1}\mathbf{X}\mathbf{C}$. If $\lambda=0$, it is easy to achieve $\epsilon=0$; otherwise, $\epsilon>0$, and a larger λ means a larger ϵ.

In summary, Algorithm 1 presents the optimization procedure to solve Eq. (5). In Algorithm 1, when $\lambda=0$, the objective value in each iteration is non-decreasing and bounded. Therefore, we have the following theorem:

Theorem 2: *When* $\lambda=0$, *the loop step in Algorithm 1 will monotonously non-decrease in each iteration and finally converge to an optima.* (Proof in the supplemental material)

In practice, when λ is small, Algorithm 1 can also converge to an optima because each subproblem has a closed form solution. For clarity, we show the convergence process with $\lambda=0.01$ in Figure 1, which shows that the objective value is close to an optima after 3 iterations and mean average precision (MAP) becomes stable after 6 iterations.

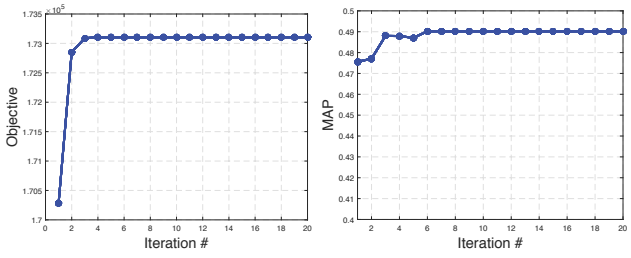
Kernel Extension

Since kernel hashing methods, which can capture the nonlinear manifold structure hidden in data, usually obtain better retrieval accuracy than linear hashing methods, we can further kernelize our graph based hashing model. Let ϕ: $\mathbb{R}^d\to\mathcal{H}$ represent a kernel mapping from the original

Algorithm 1: ADGH

Input: $\mathbf{X} \in \mathbb{R}^{d\times n}$, $\widetilde{\mathbf{X}} \in \mathbb{R}^{d\times m}$, $\mathbf{V} \in \mathbb{R}^{n\times m}$, r, γ and λ.
Output: $\mathbf{B} \in \{-1,1\}^{n\times r}$, $\mathbf{A} \in \mathbb{R}^{d\times r}$.

Initialize: t=0, let $\mathbf{A}_0$ be the eigenvectors of $\mathbf{X}\mathbf{V}\widetilde{\mathbf{X}}$ and $\mathbf{D}_0 = sgn(\widetilde{\mathbf{X}}\mathbf{A}_0)$, calculate $\mathbf{M} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}_d)^{-1}\mathbf{X}$;
Repeat
Update $\mathbf{B}_{t+1} = sgn(\mathbf{V}\mathbf{D}_t + \gamma\mathbf{X}^T\mathbf{A}_t)$;
Update $\mathbf{D}_{t+1} = sgn(\mathbf{V}^T\mathbf{B}_{t+1})$;
Update $\mathbf{C}_{t+1}$ based on Theorem 1;
Update $\mathbf{A}_{t+1} = \mathbf{M}\mathbf{C}_{t+1}$;
Until convergence



(a) @ $\lambda = 0.01$ (b) @ $\lambda = 0.01$

Figure 1: Influence of the number of iterations on retrieval performance and training time (We present the results of ADGH with 32-bit binary codes on YouTube face database with 100 classes and 30000 training data. The number (m) of selected anchors is 3000.). (a) Objective vs Iteration #, (b) MAP vs Iteration #.

space to the kernel space, where $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with a kernel function $\mathcal{K}(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})\phi(\mathbf{y})$. To map the data $\mathbf{X}$ into the kernel space, we randomly select l data points from $\mathbf{X}$ to construct a kernel matrix $\mathbf{K} \in \mathbb{R}^{l\times n}$. With normalizing the kernel matrix $\mathbf{K}$ to be $\bar{\mathbf{K}} \in \mathbb{R}^{l\times n}$ such that $\bar{\mathbf{K}}\mathbf{1}_n = 0$, we kernelize the graph based hashing model:

$$\begin{aligned} &\underset{\mathbf{B},\mathbf{D},\mathbf{A}_k}{max}\ Tr\left\{\mathbf{B}^T\mathbf{V}\mathbf{D}\right\} + \gamma Tr\left\{\mathbf{B}^T\bar{\mathbf{K}}^T\mathbf{A}_k\right\}, \\ &s.t.\ \mathbf{B} \in \{-1,1\}^{n\times r}, \mathbf{D} \in \{-1,1\}^{m\times r}, \\ &\quad \left\|\mathbf{A}_k^T\bar{\mathbf{K}}\bar{\mathbf{K}}^T\mathbf{A}_k - n\mathbf{I}_r\right\|_F^2 \le \epsilon, \end{aligned} \tag{11}$$

where $\mathbf{A}_k \in \mathbb{R}^{l\times r}$ is a projection matrix to map the kernelized data into a low-dimensional space. Since Eq. (11) is similar to Eq. (5), it is easy to solve Eq. (11) using Algorithm 1 by replacing the data matrix $\mathbf{X}$ with the kernel matrix $\bar{\mathbf{K}}$. In this paper, the kernel version of ADGH is named KADGH.

Time Complexity Analysis

In the initialization step of Algorithm 1, ADGH, the time complexity to calculate the matrices $\mathbf{A}_0$, $\mathbf{D}_0$ and $\mathbf{M}$ is $\mathcal{O}(nmd)$, $\mathcal{O}(mdr)$ and $\mathcal{O}(nd^2)$, respectively. In the loop step, calculating the matrices $\mathbf{B}_{t+1}$, $\mathbf{D}_{t+1}$, $\mathbf{C}_{t+1}$ and $\mathbf{A}_{t+1}$ requires at most $max(\mathcal{O}(nmr), \mathcal{O}(ndr))$, $\mathcal{O}(nmr)$, $\mathcal{O}(nr^2)$ and $\mathcal{O}(ndr)$ operations, respectively. Thus the time complexity of the loop step is $max(\mathcal{O}(knmr), \mathcal{O}(kndr))$, where k is the number of iterations, empirically $k \le 6$. The total complexity of Algorithm 1 is $max(\mathcal{O}(nmd), \mathcal{O}(nd^2))$, since usually $kr < m$ and $kr < d$. Similarly, the time complexity of KADGH is $max(\mathcal{O}(nml), \mathcal{O}(nl^2))$.

In the test stage, for ADGH and KADGH, the time complexity of encoding one test sample is $\mathcal{O}(dr)$ and $\mathcal{O}(ld+lr)$, respectively.

Experiment

We evaluate the proposed algorithms ADGH and KADGH on three benchmark large-scale image databases: CIFAR-10 (Torralba, Fergus, and Freeman 2008), YouTube (Wolf, Hassner, and Maoz 2011) and ImageNet (Deng et al. 2009). CIFAR-10 database is a labeled subset of 80M tiny images, constituted of 60K color images from ten object categories. Every category contains 6K images, each of which is cropped and aligned to 32×32 pixels and then represented by a 512-dimensional GIST feature vector (Oliva and Torralba 2001). YouTube face database consists of 1595 people, from which we randomly choose 100 individuals that each one has at least 310 images to form a subset for evaluation. Each face image is represented by a 1770-dimensional LBP feature vector (Ahonen, Hadid, and Pietikainen 2006). ImageNet database contains more than 1.4 million labeled images. We adopt the ILSVRC-2012 dataset, containing 1.2 million images of 1000 object categories, from which 50 categories are selected to construct a subset, and convolutional neural network (CNN) (LeCun, Bengio, and Hinton 2015) is used to extract a 4096-dimensional feature vector to represent each image.

Experimental setting

In our experiments, we split CIFAR-10 database into a training set (59K images) and a test query set (1K images), which consists of 10 categories with each containing 100 images. We also partition the selected set of YouTube face database into two parts: training and test query sets, where we randomly pick up 300 and 10 images of each individual for training and testing, respectively. For the subset of ImageNet database, we randomly choose 1000 and 20 images from each category for training and testing, respectively.

We compare ADGH and KADGH against four unsupervised methods, LSH (Indyk and Motwani 1998), SH (Weiss, Torralba, and Fergus 2009), AGH (Liu et al. 2011) and DGH (Liu et al. 2014), and five supervised methods, KSH (Liu et al. 2012), asymmetric Lin:V (Neyshabur et al. 2013), FastHash (Lin et al. 2014), SDH (Shen et al. 2015), and COSDISH (Kang, Li, and Zhou 2016). Apart from that, we also show the retrieval performance of supervised discrete graph hashing (SDGH) that preserves the symmetric discrete matrix to learn binary codes of training data (The details of the model formulation and the optimization procedure are explained in the supplemental material). We set the regularization parameter $\lambda = 0.01$ for all ADGH, KADGH and SDGH, $\gamma = 0.1$ for ADGH, $\gamma = 0.01$ for KADGH, and search the best γ during $[0.01, 100]$ for SDGH. We choose the same kernel as KSH for KADGH in experiments. To construct the affinity matrix $\mathbf{V}$, we randomly select 10 percent samples of each class as anchors for ADGH and KADGH. This process is repeated 10 times and the average

Table 1: Hamming ranking performance (%) on CIFAR-10, YouTube face and ImageNet databases (r is the number of bits of each hashing method).

Anchors	Method	CIFAR-10 MAP/Top-5900			YouTube MAP/Top-300			ImageNet MAP/Top-1000		
		$r=8$	$r=16$	$r=32$	$r=16$	$r=32$	$r=64$	$r=16$	$r=32$	$r=64$
-	LSH	17.10	17.73	18.65	6.77	11.36	19.87	12.46	19.50	28.91
	SH	17.04	17.10	16.41	22.36	29.60	35.52	29.02	32.44	36.53
	AGH_1	22.84	22.47	23.28	22.05	31.02	37.25	39.92	49.21	51.14
	AGH_2	22.45	21.72	22.39	16.37	26.70	32.81	34.26	44.14	49.17
	DGH	23.62	21.89	22.21	14.40	30.20	38.12	28.36	40.54	45.64
-	FastHash	48.66	57.81	62.90	39.20	48.00	51.00	67.98	78.10	81.24
-	COSDISH	45.70	52.65	57.55	27.30	44.26	51.70	71.73	81.00	82.85
	Lin:V	43.50	49.30	47.30	35.10	36.36	37.21	69.50	67.00	66.50
	SDGH	47.00	55.17	55.90	35.56	45.93	51.24	70.90	81.40	**83.80**
	ADGH	51.70	53.16	56.60	39.61	47.60	52.77	75.80	82.60	**83.80**
1000	KSH	37.46	43.92	47.65	36.74	45.55	48.97	52.67	62.68	66.84
	SDH	41.50	59.80	61.24	36.60	46.80	47.48	67.30	72.22	75.90
	KADGH	57.00	60.40	62.80	40.20	49.40	52.70	73.00	80.02	81.40
3000	KSH	42.89	49.09	52.11	34.46	44.05	51.09	53.83	63.23	67.52
	SDH	42.30	63.40	65.28	43.30	48.30	53.00	72.60	76.11	78.20
	KADGH	**62.50**	**64.30**	**67.50**	**44.50**	**52.38**	**55.10**	**78.40**	**82.90**	83.60

Table 2: Training and test time on CIFAR-10, YouTube face and ImageNet databases (The kernels in KSH, SDH and KADGH are constructed by 3000 anchors. r is the number of bits of each hashing method. All training and test time are in seconds.).

Method	CIFAR-10 TrainTime		CIFAR-10 TestTime	YouTube TrainTime		YouTube TestTime	ImageNet TrainTime		ImageNet TestTime
	$r=8$	$r=32$	$r=32$	$r=16$	$r=64$	$r=64$	$r=16$	$r=64$	$r=64$
LSH	0.18	0.25	3.0×10^{-6}	0.28	0.36	1.4×10^{-5}	0.49	0.64	2.0×10^{-5}
SH	0.76	1.00	1.0×10^{-5}	1.43	3.06	4.2×10^{-5}	9.38	10.95	4.0×10^{-5}
AGH_1	202.81	183.90	2.8×10^{-4}	28.51	31.67	3.3×10^{-4}	93.37	93.67	6.3×10^{-4}
AGH_2	211.33	190.91	2.7×10^{-4}	30.03	32.57	3.5×10^{-4}	104.46	105.76	6.7×10^{-4}
DGH	>10000	>10000	3.1×10^{-4}	300.76	364.16	3.3×10^{-4}	525.19	633.40	7.0×10^{-4}
FastHash	279.31	975.63	4.4×10^{-4}	221.63	768.12	8.3×10^{-4}	447.05	1290.67	8.5×10^{-4}
COSDISH	3.47	26.77	6.4×10^{-6}	6.93	52.41	1.4×10^{-5}	26.78	98.89	2.0×10^{-5}
Lin:V	>10000	>60000	1.1×10^{-4}	>20000	>50000	9.0×10^{-5}	>30000	>90000	1.8×10^{-5}
SDGH	55.09	199.44	6.4×10^{-6}	46.88	98.15	1.4×10^{-5}	215.60	885.44	2.0×10^{-5}
ADGH	6.54	6.74	6.4×10^{-6}	7.16	8.01	1.4×10^{-5}	50.04	52.09	2.0×10^{-5}
KSH	>10000	>50000	1.1×10^{-4}	>10000	>50000	1.8×10^{-4}	>10000	>50000	3.3×10^{-4}
SDH	69.40	85.89	1.1×10^{-4}	29.36	101.66	2.0×10^{-4}	55.71	145.57	3.9×10^{-4}
KADGH	35.57	37.16	1.1×10^{-4}	19.20	20.05	1.8×10^{-4}	37.21	39.23	3.3×10^{-4}

retrieval accuracy is reported. The affinity matrix is defined as:

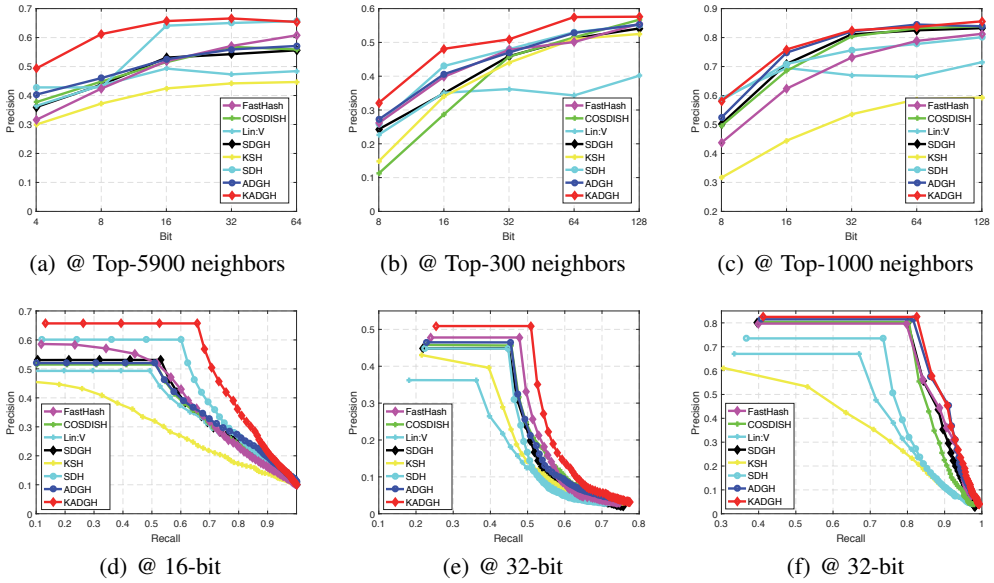
$$v_{ij} = \begin{cases} \frac{1}{n_c} & if\ \{\mathbf{x}_i, \mathbf{x}_{\mathbf{s}j}\} \in c\text{-th } class, \\ 0 & otherwise, \end{cases} \quad (12)$$

where n_c is the number of training data belonging to c-th class, $\mathbf{x}_i$ represents one training sample and $\mathbf{x}_{\mathbf{s}j}$ is the selected data.

For the kernel methods KADGH, KSH and SDH, we randomly select 1000 and 3000 samples as anchors to construct the kernels, respectively. Three standard main criterions: Precision, Mean average precision (MAP) and Precision-Recall (PR) curve, are used to evaluate the hashing methods. In addition, we also report the training and test time of each hashing method for comparison. All experiments are conducted using Matlab on a 3.50GHz Intel Xeon CPU with 128GB memory.

Experimental results

As shown in Table 1, ADGH exhibits better ranking performance than the other three linear methods, SDGH, Lin:V and COSDISH, on three databases in most cases, especially with short binary codes like 8-bit, 16-bit and 16-bit binary codes on CIFAR-10, YouTube and ImageNet databases, respectively. Compared to kernel methods SDH and KSH, KADGH exhibits better accuracy when 1000 and 3000 anchors are selected, and the gain in MAP ranges from 3.4% to 46% over the best competitor SDH on three databases with different bits. Moreover, KADGH with 3000 anchors achieves higher MAP than the other hashing methods except ADGH. Compared to unsupervised hashing methods LSH, SH, AGH_1, AGH_2 and DGH, supervised hashing methods always obtain better ranking performance on the three databases. Table 2 shows the training and testing time of different hashing methods on the three databases. In contrast to



(a) @ Top-5900 neighbors (b) @ Top-300 neighbors (c) @ Top-1000 neighbors

(d) @ 16-bit (e) @ 32-bit (f) @ 32-bit

Figure 2: Precision and PR curves of different hashing methods on three databases: (a) Precision vs. Bit on CIFAR-10, (b) Precision vs. Bit on YouTube, (c) Precision vs. Bit on ImageNet, (d) PR curve on CIFAR-10, (e) PR curve on Youtube, (f) PR curve on ImageNet.

ADGH, although COSDISH requires less training time cost on short bits (8-bit on CIFAR-10, 16-bit on Youtube and ImageNet databases), it spends much more training time on long bits (32-bit on CIFAR-10, 64-bit on Youtube and ImageNet databases). Compared to DGH, Lin:V and SDGH, ADGH consumes much less training time. Among the kernel methods, KADGH takes the least training time on all three databases, and requires a lower test time cost than SDH on YouTube and ImageNet databases. To better illustrate the retrieval performance of different supervised hashing methods, we present precision with different bits and PR-curve on three databases in Figure. 2, which indicates that KADGH outperforms the other hashing methods, and ADGH has the superior performance over SDGH and COSDISH, especially on short binary codes.

Discussion

Experiments on all the three benchmark databases demonstrate that the linear hashing method ADGH has superior retrieval performance over SDGH and COSDISH, and its kernelization KADGH can further improve the retrieval accuracy and outperform the other hashing methods. We summarize and interpret our preservations as follows:

- Compared to the unsupervised hashing methods LSH, SH, AGH_1, AGH_2 and DGH, the supervised hashing methods ADGH, SDGH, Lin:V, COSDISH, FastHash, KSH and SDH can obtain better retrieval accuracy, because they utilize the semantic information to encode each image into binary codes.
- ADGH outperforms COSDISH, probably because COSDISH divides the training data into two parts to produce binary codes, thereby losing some semantic information. ADGH takes less training time cost to produce relatively long binary code than COSDISH, since the optimization procedure in COSDISH calculates the binary codes bit by bit. ADGH achieves better retrieval performance than SDGH with short binary codes (the number of bits is smaller than the number of classes), probably because the asymmetric discrete constraint can better preserve the similarity of affinity matrix than the symmetric discrete constraint. ADGH has superior retrieval accuracy (including MAP and precision) to Lin:V, probably because it can jointly learn binary codes and the projection matrix. Moreover, ADGH consumes less training time than SDGH and Lin:V because of the asymmetric affinity matrix and its efficient optimization procedure.
- Compared to ADGH, KADGH can obtain better retrieval performance because it can capture the non-linear manifold structure hidden in the data. KADGH outperforms KSH because of directly learning binary codes. KADGH can achieve better performance than SDH, probably because SDH aims to reduce the length of binary codes to the number of classes, which suggests that it achieves the best accuracy of itself when the length of binary codes is much larger than the number of classes.

Conclusion

In this paper, we propose a novel graph bashed hashing method, asymmetric discrete graph hashing, to utilize the semantic information to encode the high-dimensional data into a set of binary codes. The proposed method preserves the asymmetric discrete constraint and builds an asymmetric affinity matrix to maintain the similarity of the original affinity matrix. Besides directly learning the binary codes of

training data, the method can simultaneously learn a low-dimensional projection matrix. Extensive experiments on three benchmark databases demonstrate that the proposed method outperforms the art-of-the-start methods with lower training time costs.

References

Ahonen, T.; Hadid, A.; and Pietikainen, M. 2006. Face description with local binary patterns: Application to face recognition. *TPAMI* 28(12):2037–2041.

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *JACM* 58(3):11.

Deng, J.; Dong, W.; Socher, R.; Li, L. J.; Li, K.; and Li, F. F. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255.

Gong, Y.; ; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI* 35(12):2916–2929.

Hou, C.; Nie, F.; Li, X.; Yi, D.; and Wu, Y. 2014. Joint embedding learning and sparse regression: A framework for unsupervised feature selection. *T Cybernetics* 44(6):793–804.

Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *ACM STOC*, 604–613.

Joly, A., and Buisson, O. 2011. Random maximum margin hashing. In *CVPR*, 873–880.

Kang, W.; Li, W.; and Zhou, Z. 2016. Column sampling based discrete supervised hashing. In *AAAI*.

Kulis, B., and Darrell, T. 2009. Learning to hash with binary reconstructive embeddings. In *NIPS*, 1042–1050.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(28):436–444.

Lin, G.; Shen, C.; Shi, Q.; Hengel, A.; and Suter, D. 2014. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 1963–1970.

Liu, W.; Wang, J.; Kumar, S.; and Chang, S. F. 2011. Hashing with graphs. In *ICML*, 1–8.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.; and Chang, S. 2012. Supervised hashing with kernels. In *CVPR*, 2074–2081.

Liu, W.; Cun, M.; Kumar, S.; and Chang, S. F. 2014. Discrete graph hashing. In *NIPS*, 3419–3427.

Neyshabur, B.; Srebro, N.; Salakhutdinov, R. R.; Makarychev, Y.; and Yadollahpour, P. 2013. The power of asymmetry in binary hashing. In *NIPS*, 2823–2831.

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

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

Recht, B.; Re, C.; Tropp, J.; and Bittorf, V. 2012. Factoring nonnegative matrices with linear programs. In *NIPS*, 1214–1222.

Shen, F.; Shen, C.; Liu, W.; and Shen, H. T. 2015. Supervised discrete hashing. In *CVPR*, 37–45.

Shi, X.; Guo, Z.; Lai, Z.; Yang, Y.; Bao, Z.; and Zhang, D. 2015. A framework of joint graph embedding and sparse regression for dimensionality reduction. *TIP* 24(4):1341–1355.

Shi, X.; Guo, Z.; Nie, F.; Yang, L.; You, J.; and Tao, D. 2016a. Two-dimensional whitening reconstruction for enhancing robustness of principal component analysis. *TPAMI* 38(10):2130–2136.

Shi, X.; Xing, F.; Cai, J.; Zhang, Z.; Xie, Y.; and Yang, L. 2016b. Kernel-based supervised discrete hashing for image retrieval. In *ECCV*, 419–433.

Strecha, C.; Bronstein, A. M.; Bronstein, M. M.; and Fua, P. 2012. Ldahash: Improved matching with smaller descriptors. *TPAMI* 34(1):66–78.

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

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

Weiss, Y.; Torralba, A.; and Fergus, R. 2009. Spectral hashing. In *NIPS*, 1753–1760.

Wolf, L.; Hassner, T.; and Maoz, I. 2011. Face recognition in unconstrained videos with matched background similarity. In *CVPR*, 529–534.

Wright, J.; Ganesh, A.; Rao, S.; Peng, Y.; and Ma, Y. 2009. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *NIPS*, 2080–2088.

Yan, S.; Xu, D.; Zhang, B.; Zhang, H.; Yang, Q.; and Lin, S. 2007. Graph embedding and extensions: a general framework for dimensionality reduction. *TPAMI* 29(1):40–51.

Zhang, D.; Wang, J.; Cai, D.; and Lu, J. 2010. Self-taught hashing for fast similarity search. In *ACM SIGIR*, 18–25.

Zhou, T.; Bian, W.; and Tao, D. 2013. Divide-and-conquer anchoring for near-separable nonnegative matrix factorization and completion in high dimensions. In *ICDM*, 917–926.