Large Graph Hashing with Spectral Rotation

Xuelong Li and **Di Hu** and **Feiping Nie**[*]
School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL),
Northwestern Polytechnical University,
Xi'an 710072, Shaanxi, P. R. China
xuelong_li@opt.ac.cn, hdui831@mail.nwpu.edu.cn, feipingnie@gmail.com

Abstract

Faced with the requirements of huge amounts of data processing nowadays, hashing techniques have attracted much attention due to their efficient storage and searching ability. Among these techniques, the ones based on spectral graph show remarkable performance as they could embed the data on a low-dimensional manifold and maintain the neighborhood structure via a non-linear spectral eigenmap. However, the spectral solution in real value of such methods may deviate from the discrete solution. The common practice is just performing a simple rounding operation to obtain the final binary codes, which could break constraints and even result in worse condition. In this paper, we propose to impose a so-called spectral rotation technique to the spectral hashing objective, which could transform the candidate solution into a new one that better approximates the discrete one. Moreover, the binary codes are obtained from the modified solution via a minimizing the Euclidean distance, which could result in more semantical correlation within the manifold, where the constraints for codes are always held. We provide an efficient alternative algorithm to solve the above problems. And a manifold learning perceptive for motivating the proposed method is also shown. Extensive experiments are conducted on three large-scale benchmark datasets and the results show our method outperforms state-of-the-art hashing methods, especially the spectral graph ones.

Introduction

With the increasing demand of massive data organization, storage, and retrieval, hashing technique has attracted universal attention at present. Hashing is mainly explored to map the raw image or document into a sequence of binary codes while preserving the similarity structure among the original data. As the generated short binary codes take a few storage space and computation cost, hashing has been widely used in machine learning and computer vision, e.g., multimodal learning (Zhang, Wang, and Si 2011), classification (Gong et al. 2013a), image retrieval (Liu et al. 2011), and image patch matching (Korman and Avidan 2011).

Unsupervised hashing methods can be roughly grouped into two categories, data-independent and data-dependent (Wang et al. 2016). The well-known hashing technique *Locality Sensitive Hashing* (LSH) (Andoni and Indyk 2006) belongs to the first one. It utilizes random projection to map the data into binary codes. And the projection can be realized by various similarity measures, e.g. p-norm distance for $p \in (0\ ,\ 1]$ (Datar et al. 2004), kernel similarity (Kulis and Grauman 2012). Although the random fashion takes low computation complexity, it need more binary codes to achieve high precision and recall (Shen et al. 2013; Liu et al. 2014), which leads to the improvement of corresponding storage space and retrieval time cost.

The second category, the data-dependent hashing method, is mainly developed to generate more compact codes and has attracted more attention in recent research. Unlike the randomly selected hyperplane in the LSH, these methods attempt to learn the parameters with different projection functions from a training set . For example, *Iterative Quantization* (ITQ) (Gong et al. 2013b) utilizes PCA projection to generate hash function with large variance, while Isotropic Hashing (Kong and Li 2012) aims to produce the projected dimensions with equal variance. And these hashing functions can also be learned in the kernel space, such as *Binary Reconstruction Embedding* (BRE) (Kulis and Darrell 2009). Besides, hashing method based on reconstruction network is also considered, e.g., *Binary Autoencoder* (BA) (Carreira-Perpinán and Raziperchikolaei 2015). However, compared with the aforementioned methods, hashing technique based on the graph Laplacian shows its effectiveness in embedding the original data into low dimensional space, i.e. the compact binary codes. This mainly results from the more semantically accurate neighbors after the nonlinear embedding of Laplacian eigenmaps (Shen et al. 2013).

This paper focuses on the unsupervised graph Laplacian hashing method. The previous hashing based on spectral graph, e.g., *Spectral Hashing* (SH) (Weiss, Torralba, and Fergus 2009) and *Anchor Graph Hashing* (AGH) (Liu et al. 2011), mainly suffers from two problems. First, the discrete codes are not exactly the spectral solution but a rounding result of the real value codes to spectral relaxation. The directly thresholding scheme may lead to the improving error when the hashing length increases. Although *Discrete Graph Hasing* (DGH) (Liu et al. 2014) proposes a discrete optimization framework to directly learn the binary codes, but it needs much more time and cannot guarantee the opti-

[*]Corresponding Author.

mal solution to the original objective. Second, the obtained real value codes from the top-K smallest eigenvector may deviate from the discrete solution, and there may exist better solution that fits the spectral relaxation. In this paper, in order to address the aforementioned two problems, we propose a method, named as *Large Graph Hashing with Spectral Rotation* (LGHSR), which applies the spectral rotation to the graph hashing method and could get a closer real value code to the discrete solution than existing results. Moreover, the final binary codes are obtained from an alterative optimization method while maintaining the constraint for codes, which results in efficient low dimensional representation and takes low time complexity. Our method is evaluated on three large-scale datasets, and shows better compact codes over state-of-the-art methods on various metrics.

In the rest of this paper, we first revisit the hashing technique based on spectral graph in Section 2. Then we propose our objective function and provide the detailed optimization algorithm in Section 3. Section 4 provides a kind of manifold learning perspective for the proposed method. Section 5 conducts different sets of experiments for evaluating it on public datasets. In the end, Section 6 concludes this paper.

Spectral Graph Hashing Revisit

Hashing aims at projecting the high dimensional real value data $\mathbf{x}_i \in R^d$ into r-bits binary codes $\mathbf{f}^i \in \{-1,1\}^r$, and keeping the similarity structure of these n datapoints. It can be formulated as the following minimization problem (Weiss, Torralba, and Fergus 2009),

$$\begin{aligned} &\min_{\mathbf{F}} \quad \sum_{i,j=1} W_{ij} \left\| \mathbf{f}^i - \mathbf{f}^j \right\|_2^2, \\ &s.t. \quad \mathbf{F} \in \{-1,1\}^{n\times r},\ \mathbf{F}^T\mathbf{F} = n\mathbf{I},\ \mathbf{F}^T\mathbf{1} = \mathbf{0} \end{aligned} \tag{1}$$

where $\mathbf{f}^i$ is the ith row of $\mathbf{F}$, and the affinity matrix $\mathbf{W}$ indicates the similarity of original data via Euclidean distance $W_{i,j} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \big/ 2\varepsilon^2)$, and ε is the bandwidth parameter. The constraints $\mathbf{F}^T\mathbf{F} = n\mathbf{I}$ enforces the different bits to be uncorrelated and $\mathbf{F}^T\mathbf{1} = \mathbf{0}$ forces each bit has equal chance to be -1 or 1. These constraints make the binary codes be more efficient for encoding the original data.

As Eq. (1) seeks the compact binary codes via minimizing the average Hamming distance between similar points, it is equivalent to solve the problem of balanced graph partitioning. Even so, the latter one is still a NP-hard problem. One general way is to relax it via spectral method, which could be written as,

$$\begin{aligned} &\min_{\mathbf{F}} \quad Tr\left(\mathbf{F}^T(\mathbf{D}-\mathbf{W})\mathbf{F}\right), \\ &s.t. \quad \mathbf{F} \in \{-1,1\}^{n\times r},\ \mathbf{F}^T\mathbf{F} = n\mathbf{I},\ \mathbf{F}^T\mathbf{1} = \mathbf{0} \end{aligned} \tag{2}$$

where $(\mathbf{D}-\mathbf{W})$ is the graph Laplacian (Chung 1997), $\mathbf{D}$ is a diagonal matrix with the element $d_i = \sum_j W_{ij}$, and Tr denotes the matrix trace operation. However, the first discrete constraint still makes Eq. (2) hard to be solved. A kind of spectral relaxation could turn it into an easy problem by removing this constraint, where the solution becomes the r eigenvectors of the graph Laplacian with minimal eigenvalue except the eigenvalue of zero. For a novel data out of the training set, it is assumed that the datapoint is sample from a uniform distribution, which could therefore obtain an analytical solution (Weiss, Torralba, and Fergus 2009).

However, hashing technique requires efficient coding, the time complexity of constructing affinity matrix $\mathbf{W}$ is $O\left(dn^2\right)$. When the training samples and feature dimension become very large, e.g. CIFAR-10 has 60K images and each sample has 512D GIST feature (Krizhevsky and Hinton 2009), a big training time consuming has to be faced with. To solve this problem, AGH considers an approximation of the matrix $\mathbf{W}$ by only preserving the neighborhood structure of each data point (Liu, He, and Chang 2010). Specifically, a small set of m points are obtained through clustering on the training data n, where $m \ll n$. The sets of points are treated as the anchors, $\mathcal{U} = \left\{\mathbf{u}_j \in R^d | j = 1, \ldots, m\right\}$, to approximate the neighborhood structure. Then a nonlinear mapping from data to anchor is computed as follows,

$$Z_{ij} = \begin{cases} \frac{K(\mathbf{x}_i,\mathbf{u}_j)}{\sum_{\hat{j}\in\langle i\rangle} K\left(\mathbf{x}_i,\mathbf{u}_{\hat{j}}\right)}, & \forall j \in \langle i\rangle \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $\langle i\rangle$ indicates the s nearest anchors around $\mathbf{x}_i$ ($s \ll m$), the function $K(\cdot)$ is used to measure the similarity between data $\mathbf{x}_i$ and anchor $\mathbf{u}_j$ with ℓ_2 distance in Gaussian kernel space $K(\mathbf{x}_i, \mathbf{u}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{u}_j\|_2^2 \big/ 2\sigma^2\right)$, and σ is the bandwidth parameter. The normalized matrix $\mathbf{Z} \in R^{n\times m}$ is then utilized to obtain a low-rank affinity matrix $\mathbf{A} \in R^{n\times n}$,

$$\mathbf{A} = \mathbf{Z}\mathbf{\Lambda}^{-1}\mathbf{Z}^{\mathrm{T}}, \tag{4}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\mathbf{Z}^T\mathbf{1})$. The construction of affinity matrix can also be accelerated via (Nie, Zhu, and Li 2017). As the $\mathbf{\Lambda}^{-1}$ normalizes the constructed matrix $\mathbf{A}$, the summation of each column and row equal to one and the graph Laplacian becomes $\mathbf{I} - \mathbf{A}$. Then, the Eq. (2) can be re-written as a maximization problem,

$$\begin{aligned} &\max_{\mathbf{F}} \quad Tr\left(\mathbf{F}^T\mathbf{A}\mathbf{F}\right). \\ &s.t. \quad \mathbf{F} \in \{-1,1\}^{\mathrm{n\times r}},\ \mathbf{F}^{\mathrm{T}}\mathbf{F} = \mathrm{n}\mathbf{I},\ \mathbf{F}^{\mathrm{T}}\mathbf{1} = \mathbf{0} \end{aligned} \tag{5}$$

Similar with Eq.(2), the first constraint should also be removed, which turns Eq. (5) into

$$\begin{aligned} &\max_{\mathbf{F}} \quad Tr\left(\mathbf{F}^T\mathbf{A}\mathbf{F}\right). \\ &s.t. \quad \mathbf{F}^T\mathbf{F} = n\mathbf{I},\ \mathbf{F}^{\mathrm{T}}\mathbf{1} = \mathbf{0} \end{aligned} \tag{6}$$

The solution to this problem becomes the r eigenvectors of the matrix $\mathbf{A}$ with maximal eigenvalue (except the eigenvector which has eigenvalue one). Such real value code results are often rounded into binary codes via a hard threshold of zero (Weiss, Torralba, and Fergus 2009; Liu et al. 2011; Shen et al. 2013).

Recently, the discrete constraint $\mathbf{F} \in \{-1,1\}^{n\times r}$ is considered again. But a kind of extra penalty term is combined with the original problem (Eq. 5), which is a distance between the binary code solution and a real value code set (Liu et al. 2014). When the hyper-parameter of the penalty term becomes very large, the proposed problem will turn into the original one and cost a lot of time to converse. But, if the parameter becomes small, the obtained solution may be away from the solution to Eq. (5) and result in a poor binary code.

The aforementioned weaknesses of the hashing methods based on spectral graph can be grouped into two categories. First, these methods almost tend to take advantage of the spectral relaxation to obtain a real value solution, and then directly threshold the solution into a binary code. But the rounding operation may result in improving error as the code length r increases and even breaking the last two constraints. Second, there is no guarantee that such yielded real value solution best approximates the binary codes. As a matter of fact, there have been numerous solutions to Eq. (6) and we could find a closer real value solution to the discrete one.

Spectral Rotation Meets Better Solution

The spectral rotation technique has the property of spectral solution invariance, which could provide a solution set to Eq. 6 and find a better real value solution to the discrete codes. Although the technique has been applied to optimize the objective function in spectral clustering, such as (Zelnik-Manor and Perona 2005; Nie et al. 2011; Huang, Nie, and Huang 2013), it is barely noticed in the hashing method. In this paper, we aim at applying the spectral rotation to the spectral graph hashing method and seeking the real value solution and binary codes in an iterative manner.

Let us first solve the spectral relaxation problem (Eq. (6)) to obtain a candidate real value solution $\mathbf{F}$. In fact, for any $\mathbf{F}$, $\mathbf{FQ}$ is another solution, where $\mathbf{Q}$ is an arbitrary orthonormal matrix. Hence, we hope we can find a better $\mathbf{FQ}$ that is closer to the binary codes $\mathbf{B}$, which can be formulated as,

$$\begin{aligned} \min_{\mathbf{F}} \quad & \|\mathbf{B}-\mathbf{FQ}\|_F^2, \\ s.t. \quad & \mathbf{B}\in\{-1,1\}^{n\times r}, \mathbf{B}^T\mathbf{1}=\mathbf{0}, \mathbf{Q}^T\mathbf{Q}=\mathbf{I} \end{aligned} \tag{7}$$

where the distance from the discrete solution $\mathbf{B}$ to the modified real value solution $\mathbf{FQ}$ is measured in terms of ℓ_2 metric. Different from the constraint violation of $\mathbf{B}$ in the previous methods, Eq. (7) still preservers the constraint that requires each bit to be 50% of time and just relaxes the uncorrelated constraint, which could make the code be more efficient.

The objective function Eq. (7) can be solved by employing iteratively alternative minimization.

When $\mathbf{Q}$ is fixed and $\mathbf{M}=\mathbf{FQ}$, Eq. (7) can be re-written as,

$$\begin{aligned} \min_{\mathbf{B}} \quad & \|\mathbf{B}-\mathbf{M}\|_F^2 = \sum_j \|\mathbf{b}_j-\mathbf{m}_j\|_2^2. \\ s.t. \quad & \mathbf{B}\in\{-1,1\}^{n\times r}, \mathbf{B}^T\mathbf{1}=\mathbf{0} \end{aligned} \tag{8}$$

Hence, it is equivalent to minimizing the ℓ_2-*norm* of each column of $\mathbf{B}-\mathbf{M}$. First, let $\mathbf{b}$ and $\mathbf{m}$ denote the column vectors instead of $\mathbf{b}_j$ and $\mathbf{m}_j$ for simplification, respectively. Then, due to $\mathbf{b}\in\{-1,1\}^{n\times 1}$, the minimization of each column vector subtraction is also equivalent to the following equation,

$$\begin{aligned} \max_{\mathbf{b}} \quad & \mathbf{b}^T\mathbf{m}. \\ s.t. \quad & \mathbf{b}\in\{-1,1\}^{n\times 1}, \mathbf{b}^T\mathbf{1}=0 \end{aligned} \tag{9}$$

The maximization of Eq. (9) can be directly solved by sorting the elements of $\mathbf{m}$ in descending order, and then assigning 1 or -1 according to the sorted result:

$$\mathbf{b}_{\mathbf{i}_k} = \begin{cases} 1, & k\le n/2 \\ -1, & \text{otherwise} \end{cases} \tag{10}$$

where $\mathbf{i}$ is the index vector of the elements in $\mathbf{v}$ after the sorting. The assigned vector $\mathbf{b}$ finally constitutes the matrix $\mathbf{B}$ as a column.

When $\mathbf{B}$ is fixed, Eq. (7) can be re-written as,

$$\begin{aligned} \max_{\mathbf{Q}} \quad & Tr(\mathbf{G}^T\mathbf{Q}), \\ s.t. \quad & \mathbf{Q}^T\mathbf{Q}=\mathbf{I} \end{aligned} \tag{11}$$

where $\mathbf{G}=\mathbf{F}^T\mathbf{B}$. And the we can obtain an analytical solution to Eq. (11) via the Theorem 1.

Theorem 1 *$\boldsymbol{Q}=\boldsymbol{U}\boldsymbol{V}^T$ is the optimal solution to the problem in Eq. (11), where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left- and right-part of the compact Singular Value Decomposition (SVD) of $\boldsymbol{G}$.*

Proof

First, it is obvious that $\mathbf{Q}=\mathbf{U}\mathbf{V}^T$ is feasible to Eq. (11).

Then, for an arbitrary $\mathbf{Q}$, suppose that the compact SVD of $\mathbf{G}$ is $\mathbf{G}=\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, Eq. (11) can be re-written as,

$$\begin{aligned} Tr(\mathbf{G}^T\mathbf{Q}) &= Tr\left(\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^T\mathbf{Q}\right) && (12) \\ &= Tr\left(\boldsymbol{\Gamma}\boldsymbol{\Sigma}\right), && (13) \end{aligned}$$

where $\boldsymbol{\Gamma}=\mathbf{U}^T\mathbf{Q}\mathbf{V}$. According to the von Neumann's trace inequality (Mirsky 1975),

$$Tr\left(\boldsymbol{\Gamma}\boldsymbol{\Sigma}\right) \le \sum_{i=1}^{r} \rho_i\sigma_i, \tag{14}$$

where $\rho_1\ge\cdots\ge\rho_r$ and $\sigma_1\ge\cdots\ge\sigma_r$ are the singular values of $\boldsymbol{\Gamma}$ and $\boldsymbol{\Sigma}$, respectively. As $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T=\mathbf{I}$, the singular values $\rho_i=1$ and Eq. (14) becomes $Tr\left(\boldsymbol{\Gamma}\boldsymbol{\Sigma}\right)\le\sum_{i=1}^{r}\sigma_i$. The equality holds when $\boldsymbol{\Gamma}=\mathbf{I}$, which leads to

$$\mathbf{Q}=\mathbf{U}\mathbf{V}^T. \tag{15}$$

This completes our proof. □

The proposed optimization of $\mathbf{B}$ and $\mathbf{Q}$ are executed iteratively until satisfying the convergence criteria, i.e. the unchanged binary code matrix $\mathbf{B}$. We summarize the proposed LGHSR in Algorithm 1. The time complexity of the algorithm is $O\left(r^2nN+rnN\log_2 n\right)$, where N is the budget iteration number. Note that the former part corresponds to solving $\mathbf{Q}$ and the latter is for solving $\mathbf{B}$. As the code length r is a tiny number compared with n, it can be expected that the method will be achieved in a few seconds.

Out-of-Sample

For a new coming data $\mathbf{x}\in R^d$, the corresponding binary codes should also be derived. In consideration of the caused errors of rounding operation in generating codes for training data, we employ the similar objective as the training procedure but use the real value solution instead of the discrete one for improving robustness,

$$\min_{\mathbf{b}(\mathbf{x})\in\{-1,1\}^r} \quad \sum_{i=1}^{n}\mathbf{A}\left(\mathbf{x}_i,\mathbf{x}\right)\|\mathbf{m}_i-\mathbf{b}\left(\mathbf{x}\right)\|_2^2. \tag{16}$$

Algorithm 1 Large Graph Hashing with Spectral Rotation

```
Input: Training data $\mathbf{X} \in R^{n\times d}$, code length $r$, anchor size $m$, neighbor number $s$, and number of iteration $N$.
Output: The binary codes $\mathbf{B} \in R^{n\times r}$.
1: Build the anchor graph and compute the approximated affinity matrix $\mathbf{A}$ by Eq. (3) and Eq. (4), respectively.
2: Obtain the candidate real value solution $\mathbf{F}$ via solving Eq. (6).
3: Initial an arbitrary orthogonal matrix $\mathbf{Q}$.
4: Fix $\mathbf{Q}$, assign 1 or $-1$ to the elements of each column of $\mathbf{B}$ by Eq. (10).
5: Fix $\mathbf{B}$, update $\mathbf{Q}$ and obtain better real value solution by Eq. (15).
6: Repeat above 4-5 steps until convergence or $N$ steps.
```

As $\mathbf{A}(\mathbf{x}_i, \mathbf{x}) = \mathbf{z}^i \mathbf{\Lambda}^{-1} \mathbf{z}(\mathbf{x})$, where $\mathbf{z}^i$ is the i-row of matrix $\mathbf{Z}$, Eq. (16) is equivalent to,

$$\max_{\mathbf{b}(\mathbf{x})\in\{-1,1\}^r} \left\langle \mathbf{b}(\mathbf{x}), (\mathbf{FQ})^T \mathbf{Z}\mathbf{\Lambda}^{-1}\mathbf{z}(\mathbf{x}) \right\rangle . \tag{17}$$

As $\mathbf{b}(\mathbf{x})$ is restricted to be binary value, the optimal solution to Eq. (17) becomes

$$\mathbf{b}(\mathbf{x}) = \operatorname{sgn}(\mathbf{Pz}(\mathbf{x})), \tag{18}$$

where $\operatorname{sgn}(\cdot)$ is the sign function and $\mathbf{P} = (\mathbf{FQ})^T \mathbf{Z}\mathbf{\Lambda}^{-1}$ can be pre-computed, which could make the out-of-sample hashing more effective.

Manifold Learning Perspective of Spectral Rotation

The real value codes obtained by the Laplacian eigenmap can be considered as the points embedded on a low-dimensional manifold via solving the spectral relaxation problem in Eq. (6). As shown in Fig. 1, the neighborhood structure of the high dimensional data points are preserved in the manifold. However, the embedded points are still away from the corresponding discrete codes. And directly carrying out the rounding operation may even confuse the binary codes that belong to non-adjacent data points. In this condition, spectral rotation takes advantage of the orthogonal matrix $\mathbf{Q}$, which could still derive the optimal spectral solution and remain the neighborhood structure among the data points. It is employed for finding better solution that is closer to the corresponding discrete codes (in terms of ℓ_2), which is considered to be a kind of transformation, i.e. the red arrow in Fig. (1). Hence, the transformed solution should be more reliable for generating the discrete codes.

To verify the effectiveness of spectral rotation, it is performed with the real value solution of AGH (Liu et al. 2011) and compared with the corresponding discrete solution. Note that, as AGH directly rounds the real value solution via zero, it is fair to perform the similar operation in the spectral rotation. Hence, the second constraint in Eq. (7) is not considered, and the rounding operation is adopted for updating $\mathbf{B}$. Multiple anchor number m are considered, as it plays an important role on building the affinity matrix $\mathbf{A}$

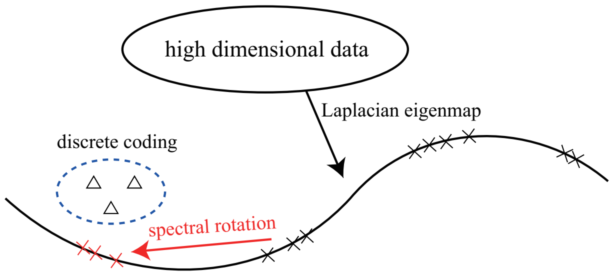

Figure 1: Manifold learning perspective for spectral rotation. Suppose the training data are embedded into the low-dimensional manifold and represented by the adjacent cross points in black. The cross points in red represent the obtained solution via the proposed the spectral rotation. And the triangles in blue ellipse represent parts of the discrete codes of the training data.

Code Length	MAP	100	300	500	700	1000
48 *bits*	AGH	17.89	18.68	18.67	19.29	19.41
	AGH+SR	18.12	20.65	21.07	20.84	20.71
96 *bits*	AGH	15.07	17.66	16.46	16.29	16.55
	AGH+SR	16.46	19.82	20.88	20.56	20.51

Table 1: Comparison between AGH and the AGH with *Spectral Rotation* (SR) on CIFAR-10, where the anchor number m varies from 100 to 1000.

(Liu, He, and Chang 2010). Table 1 displays the comparison results in *Mean Average Precision* (MAP). The MAP of AGH after performing the spectral rotation have a big improvement on both 48 *bits* and 96 *bits* codes. This shows that the spectral rotation makes the real value solution better approximate the discrete one, as shown in Fig. 1. What's more, when the code length becomes larger, the performance of AGH reduces remarkably but the spectral rotation holds to some extent. We consider that the improper embedded points by AGH make final discrete codes confused and result in the increasing error, while the modified points can overcome it by transforming the points into proper positions.

Experiments

There are three large-scale datasets used for evaluating the above methods, MNIST[1] (LeCun et al. 1998), CIFAR-10[2] (Krizhevsky and Hinton 2009), and YouTube Faces[3] (Wolf, Hassner, and Maoz 2011). The well-known MNIST dataset consists of 70,000 images associated with digits from 0 to 9, each of 784-dimensions. It is split into a training set of 69,000 samples and a testing set of 1,000 samples (100 samples for each digit). The CIFAR-10 dataset consists of

[1]http://yann.lecun.com/exdb/mnist/
[2]http://www.cs.toronto.edu/~kriz/cifar.html
[3]http://www.cs.tau.ac.il/~wolf/ytfaces/

60,000 32×32 color images, with 6,000 images per object category. Each images is represented by a 512-dimensional GIST feature vector (Oliva and Torralba 2001). The testing set consists of 100 images of each category, which are uniformly randomly sampled. And the training set is with all remaining samples. The YouTube Faces dataset contains 3,425 videos of 1,595 different people. Following the settings in (Liu et al. 2014), we form a new subset that consists of 370,319 face images belong to 340 people, where each people has at least 500 images. A 1,770-dimensional LBP feature vector (Ahonen, Hadid, and Pietikainen 2006) is utilized to represent each face image. 100 face images are uniformly sampled from the people category that contains more than 2,000 images, and the final sampled 3,800 images from 38 people form the test set. The remaining samples in the built subset form the training set. The samples with the same label are considered as the ground truth for evaluating the retrieval result for each dataset.

The proposed method is compared with several unsupervised hashing methods, including LSH (Andoni and Indyk 2006), *Kernel LSH* (KLSH) (Kulis and Grauman 2012), ITQ (Gong et al. 2013b), BRE (Kulis and Darrell 2009), BA (Carreira-Perpinán and Raziperchikolaei 2015), SH (Weiss, Torralba, and Fergus 2009), *Scalable Graph Hashing* (SGH) (Jiang and Li 2015), *Inductive Manifold Hashing* (IMH) (Shen et al. 2013), DGH (DGH-R) (Liu et al. 2014), and AGH (Liu et al. 2011). Note that, these methods cover the data-independent and data-dependent category, and the last five belong to the spectral graph based hashing methods. Considering the common setting of anchor number m and neighborhood number s (Shen et al. 2013; Liu et al. 2014), we set $m = 300$ and $s = 3$ in the experiments. And K-means is utilized to generate the anchor points of the training data. The iteration N is set to 20 for all the experiments. Besides, Hamming ranking and hash lookup are both employed for the evaluation of above methods. Specifically, *Mean Average Precision* (MAP) is computed based on the Hamming distance to a query for the Hamming ranking, and the hash lookup performance is according to a Hamming ball of radius 2 to a query.

Results on MNIST. Fig. (2) shows the comparison results on MNIST dataset in MAP and F-measure (for hash lookup). We can see that our proposed method outperforms the other

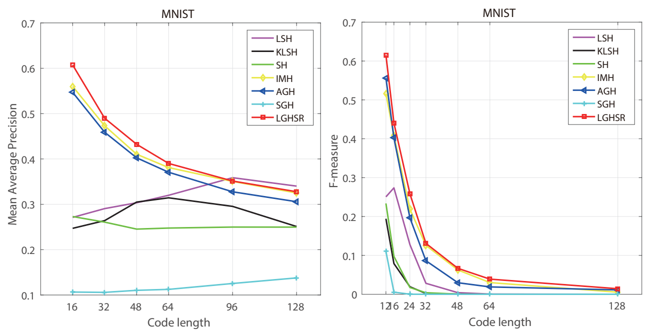

Figure 2: Hamming ranking (in MAP) and hash lookup (in F-measure) performance on MNIST.

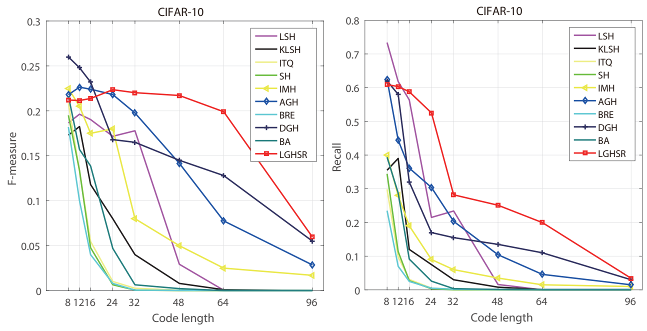

Figure 3: Hash lookup performance (in F-measure and recall) of different hashing methods on CIFAR-10.

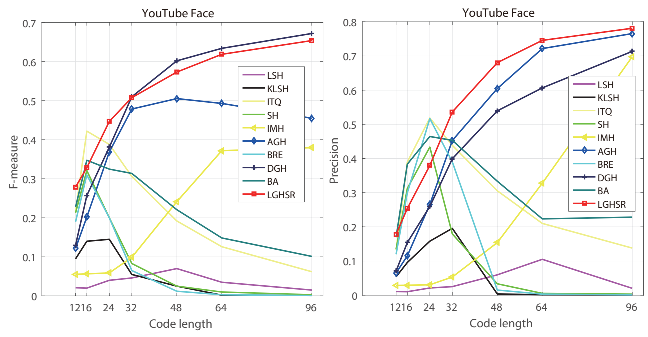

Figure 4: Hash lookup performance (in F-measure and precision) of different hashing methods on YouTube Face.

methods with almost all the code lengths, except the largest one. To be specific, LGHSR with short codes has a remarkable improvement in both MAP and F-measure, which results from the effectiveness of spectral rotation. However, when the code length becomes large, LGHSR do not hold the high performance, even worse than SGH in MAP and AGH in F-measure. This is because the constrains for codes have to be preserved in LGHSR while the threshold strategy could destroy them to keep the original low-dimensional embedding in real value. Hence, the preserved constraints could lead to a sub-optimal result compared with those methods without constraints, especially for the longer code. Even so, LGHSR still performs better than the other methods in most conditions, and the generated codes could be much more efficient as the maintained constraints.

Results on CIFAR-10. The hash lookup results in F-measure and recall are shown in Fig. (3). Compared with other methods, LGHSR achieves superior performance in terms of both metrics. Although most of the methods decrease sharply with the increasing code length, which results from the more sparse hamming space, LGHSR still remains substantial superiority over them. Note that when the code length is nearly 96 $bits$, our method just has a comparable result due to the same condition in MNIST dataset. Besides, we show the Hamming ranking performance in Table 2. It is obvious that LGHSR has a remarkable improvement over other methods with all the code length. In details, the hashing methods based on spectral graph have significant advan-

Mean Average Precision	CIFAR-10				YouTube Faces		
Code Length	24 $bits$	32 $bits$	48 $bits$	96 $bits$	48 $bits$	96 $bits$	128 $bits$
LSH	0.1349	0.1309	0.1339	0.1373	0.1241	0.1374	0.139
KLSH	0.1315	0.1394	0.1524	0.1591	0.4134	0.5210	0.6190
BA	0.1793	0.1837	0.1863	0.1902	0.6821	0.7274	0.7901
SH	0.1341	0.1324	0.1368	0.1298	0.6229	0.6789	0.7267
ITQ	0.1764	0.1757	0.1797	0.1842	0.6674	0.7135	0.7729
IMH	0.1916	0.1995	0.1889	0.1888	0.4047	0.6886	0.7422
BRE	0.1542	0.1624	0.1731	0.1823	0.5731	0.6304	0.6721
DGH	0.1910	-	0.1912	0.1950	0.7245	0.7672	0.7805
AGH	0.1907	0.1857	0.1868	0.1766	0.6759	0.7477	0.7629
LGHSR	**0.2049**	**0.2039**	**0.2033**	**0.1987**	**0.7613**	**0.8091**	**0.8428**

Table 2: Hamming ranking performance on CIFAR-10 and YouTube Faces dataset with varying code lengths.

Dataset	CIFAR-10		YouTube Faces	
Time (s)	Train	Test	Train	Test
LSH	0.8	1.4×10^{-5}	8.4	2.2×10^{-5}
BA	732.7	6.4×10^{-5}	6493.3	9.1×10^{-5}
SH	11.5	1.3×10^{-4}	121.5	2.3×10^{-4}
IMH	18.9	3.1×10^{-5}	113.7	2.6×10^{-5}
BRE	1075.4	6.1×10^{-5}	10426.3	1.0×10^{-4}
AGH	18.3	5.8×10^{-5}	114.9	4.4×10^{-5}
DGH	89.2	3.9×10^{-5}	431.6	3.2×10^{-5}
LGHSR	29.8	3.8×10^{-5}	220.2	2.7×10^{-5}

Table 3: Comparison of training and testing times (in seconds) on CIFAR-10 and YouTube Faces dataset.

tages compared with others, which confirms the effectiveness of non-linear embedding. Further, LGHSR significantly increases the MAP scores among the spectral graph methods, which states the real value solution better approximates the discrete one. What's more, the binary codes should be more efficient as the held constraints.

Results on YouTube Faces. Different from CIFAR-10 and MNIST, the number of categories in testing set is smaller than the training set of YouTube Faces dataset, which could be more difficult to retrieve for a given query. Fig. (4) shows the hash lookup performance in terms of F-measure and precision. LGHSR outperforms most of the methods but is comparable to DGH in F-measure when faced with longer code length. Nevertheless, the hyper-parameter of DGH has to be tuned to control the impact of penalty term to the spectral relaxation problem for each dataset, which also results in the relaxed constraints of learned hashing codes. In fact, our proposed method has no parameters to be tuned but keep a distinct improvement over DGH and other methods in precision. We also shows the Hamming ranking performance in Table 2, where we use the mean precision of top-retrieved samples of 2,000 instead of MAP. LGHSR shows the highest Hamming ranking results. Specifically, our method achieves a noticeable average improvement of nearly 5 points compared with DGH and much better than other methods. Besides, LGHSR keeps obvious improvements with the increasing code length, even in the sparse hamming space.

We also report the training and testing time of these methods on CIFAR-10 and YouTube Faces in Table 3. The experiments are conducted on desktop PC with a 4-core 3.20GHZ CPU and 16G RAM. Note that, the code lengths for CIFAR-10 and YouTube Faces are 96 $bits$, and 128 $bits$, respectively. It is obvious that LSH is the fastest while BRE costs the most time. Due to the spectral rotation procedure, LGHSR performs worse than the methods based on the rounding operation, i.e. AGH, IMH. However, we have a great improvement compared with them in the performance. LGHSR shows remarkable efficiency compared with the state-of-the-art methods, i.e. DGH and BA.

Conclusion

In this paper, we propose to employ spectral rotation to transform the original solution to spectral relaxation of hashing problem, which could provide a more efficient approximation to the discrete binary codes in terms of ℓ_2 distance. To maintian the constraints for binary codes, we consider a ranking method instead of the rough rounding operation, which could make the codes be more efficient. We also introduce a tractable algorithm which could divide the proposed objective into two subproblems and solves them via an alternating fashion. Moreover, a manifold learning insights is provided to explain the improvements of such method, and corresponding experiments are also conducted to prove that. Extensive experimental results on three large-scale benchmark datasets demonstrate that our proposed method can generate better compact codes and cost comparable training and testing time with most of other methods.

Acknowledgement

This work is supported by the Fundamental Research Funds for the Central Universities (Grant no. 3102015BJ(II)JJZ01).

References

Ahonen, T.; Hadid, A.; and Pietikainen, M. 2006. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence* 28(12):2037–2041.

Andoni, A., and Indyk, P. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 459–468. IEEE.

Carreira-Perpinán, M. A., and Raziperchikolaei, R. 2015. Hashing with binary autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 557–566.

Chung, F. R. 1997. *Spectral graph theory*, volume 92. American Mathematical Soc.

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, 253–262. ACM.

Gong, Y.; Kumar, S.; Rowley, H. A.; and Lazebnik, S. 2013a. Learning binary codes for high-dimensional data using bilinear projections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 484–491.

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

Huang, J.; Nie, F.; and Huang, H. 2013. Spectral rotation versus k-means in spectral clustering. In *AAAI*.

Jiang, Q.-Y., and Li, W.-J. 2015. Scalable graph hashing with feature transformation. In *Proceedings of IJCAI*.

Kong, W., and Li, W.-J. 2012. Isotropic hashing. In *Advances in Neural Information Processing Systems*, 1646–1654.

Korman, S., and Avidan, S. 2011. Coherency sensitive hashing. In *2011 International Conference on Computer Vision*, 1607–1614. IEEE.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images.

Kulis, B., and Darrell, T. 2009. Learning to hash with binary reconstructive embeddings. In *Advances in neural information processing systems*, 1042–1050.

Kulis, B., and Grauman, K. 2012. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(6):1092–1104.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Liu, W.; Wang, J.; Kumar, S.; and Chang, S.-F. 2011. Hashing with graphs. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 1–8.

Liu, W.; Mu, C.; Kumar, S.; and Chang, S.-F. 2014. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, 3419–3427.

Liu, W.; He, J.; and Chang, S.-F. 2010. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 679–686.

Mirsky, L. 1975. A trace inequality of john von neumann. *Monatshefte für Mathematik* 79(4):303–306.

Nie, F.; Zeng, Z.; Tsang, I. W.; Xu, D.; and Zhang, C. 2011. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks* 22(11):1796–1808.

Nie, F.; Zhu, W.; and Li, X. 2017. Unsupervised large graph embedding. In *AAAI*.

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

Shen, F.; Shen, C.; Shi, Q.; Van Den Hengel, A.; and Tang, Z. 2013. Inductive hashing on manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1562–1569.

Wang, J.; Liu, W.; Kumar, S.; and Chang, S.-F. 2016. Learning to hash for indexing big datała survey. *Proceedings of the IEEE* 104(1):34–57.

Weiss, Y.; Torralba, A.; and Fergus, R. 2009. Spectral hashing. In *Advances in neural information processing systems*, 1753–1760.

Wolf, L.; Hassner, T.; and Maoz, I. 2011. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 529–534. IEEE.

Zelnik-Manor, L., and Perona, P. 2005. Self-tuning spectral clustering.

Zhang, D.; Wang, F.; and Si, L. 2011. Composite hashing with multiple information sources. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 225–234. ACM.