Semi-Supervised Discriminant Hashing

Saehoon Kim
Department of Computer Science
Pohang University of Science and Technology
Pohang, Korea
Email: kshkawa@postech.ac.kr

Seungjin Choi
Deparment of Computer Science
Division of IT Convergence Engineering
Pohang University of Science and Technology
Pohang, Korea
Email: seungjin@postech.ac.kr

***Abstract*—Hashing refers to methods for embedding high-dimensional data into a similarity-preserving low-dimensional Hamming space such that similar objects are indexed by binary codes whose Hamming distances are small. Learning hash functions from data has recently been recognized as a promising approach to approximate nearest neighbor search for high-dimensional data. Most of 'learning to hash' methods resort to either unsupervised or supervised learning to determine hash functions. Recently semi-supervised learning approach was introduced in hashing where pairwise constraints (must-link and cannot-link) using labeled data are leveraged while unlabeled data are used for regularization to avoid over-fitting. In this paper we base our semi-supervised hashing on linear discriminant analysis, where hash functions are learned such that labeled data are used to maximize the separability between binary codes associated with different classes while unlabeled data are used for regularization as well as for balancing condition and pairwise decorrelation of bits. The resulting method is referred to as *semi-supervised discriminant hashing* (SSDH). Numerical experiments on MNIST and CIFAR-10 datasets demonstrate that our method outperforms existing methods, especially in the case of short binary codes.**

***Keywords*-Hashing, regularized discriminant analysis, semi-supervised learning.**

I. Introduction

Similarity search, which involves finding nearest neighbors of a query, is a core problem in various areas, including machine learning, computer vision, information retrieval, and data mining to name a few. Especially, ever-increasing availability of image data on the Web entails the need of scalable search of relevant images. For instance, content-based image retrieval (CBIR) takes an image as a query and returns its nearest neighbors, computing similarity between image descriptors (features) of the query and of an image in database. A naive solution to nearest neighbor search is linear scan where all items in database are sorted according to their similarity to the query, requiring linear complexity. However, this approach is not scalable in practical applications.

Approximate similarity search has been studied to handle the scalability, where we trade accuracy for computational speed-up. In earlier work [1], [2], spatial partitions of data space was exploited via various tree structures. While tree-based space partition approach is successful for low-dimensional data, its performance is not satisfactory for high-dimensional data and does not guarantee faster search compared to linear scan [3]. In the case of CBIR, image descriptors (such as SIFT [4] and GIST [5]) constitute high-dimensional vectors, so tree-based space partition approach is not preferred in such applications.

Hashing refers to methods for embedding high-dimensional data into a similarity-preserving low-dimensional Hamming space such that similar objects are indexed by binary codes whose Hamming distances are small. Hashing is known to be better suited to approximate similarity search for high-dimensional data. A notable data-independent method is *locality sensitive hashing* (LSH) [3] where random projections followed by rounding are used to generate binary codes such that two objects in database within a small distance are shown to have a higher probability of collision (i.e., having the same hash code). While LSH was successfully applied to the task of image retrieval, the performance is degraded when short binary codes (small number of hash functions) are used [6]. In other words, the performance of LSH increases with more hash functions, but it would be desirable to learn a compact code for large scale image retrieval applications. Thus, data-dependent hashing methods have drawn attractions recently, where binary codes are learned from data in unsupervised manner or from labeled examples in supervised fashion. An exemplary unsupervised hashing method is spectral hashing (SH) [7] where a subset of eigenvectors of the Laplacian of the similarity graph is rounded to determine binary codes. Semantic hashing [8] uses multi layers of restricted Boltzman machines to learn a non-linear mapping between input data and binary code bits. Parameter-sensitive hashing [9], which is an extension of LSH, learns an embedding of input data space into a new space where distances between data points are computed using a weighted Hamming distance and weights are learned by Boosting. In general, supervised hashing methods are slower than unsupervised methods and are easily over-fitted when only small number of labeled examples are available.

Recently semi-supervised hashing (SSH) method [10] was proposed, in which where an empirical error which measures the violation of pairwise constraints (must-link and cannot-link) is minimized using labeled data while variance and independence of hash bits are maximized over labeled and unlabeled data. Most of data-dependent hashing methods learn a compact binary code from a set of training data but separability between short binary codes were not considered. In this paper we base our semi-supervised hashing on linear discriminant analysis to learn discriminative binary codes, where hash functions are learned such that labeled data are used to maximize the *separability* between binary codes associated with different classes while unlabeled data are used for regularization as well as for balancing condition and pairwise decorrelation of bits. The resulting method is referred to as *semi-supervised discriminant hashing* (SSDH). Numerical experiments on MNIST and CIFAR datasets demonstrate that our method generates more discriminant binary codes, improving the performance in the task of CBIR, especially when short binary codes are used, compared to existing hashing methods.

II. Related Work

We briefly review three representative hashing methods, including LSH [3], spectral hashing [7], and semi-supervised hashing [10]. Suppose that we have N data points, so the data matrix is given by $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ where each data point $\boldsymbol{x}_i \in \mathbb{R}^D$ is a D-dimensional vector. Without loss of generality, the first N_l data points are labeled examples assigned one of K classes. Thus the data matrix consists of labeled examples and unlabeled examples, i.e., $\boldsymbol{X} = [\boldsymbol{X}_l, \boldsymbol{X}_u]$ where $\boldsymbol{X}_l \in \mathbb{R}^{D \times N_l}$ contains N_l labeled examples and $\boldsymbol{X}_u \in \mathbb{R}^{D \times N_u}$ has $N_u = N - N_l$ unlabeled examples. When we need to specify class labels for data, we use $\boldsymbol{x}_i^{(k)}$, which implies that $\boldsymbol{x}_i$ belongs to class k. The size of class k (the number of samples in class k) is denoted by N_k. Hashing seeks binary codes $\boldsymbol{y}_i \in \{+1, -1\}^M$ associated with $\boldsymbol{x}_i$ for $i = 1, \ldots, N$, such that Hamming distance between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ is small if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are semantically similar.

A. Locality Sensitive Hashing

The underlying idea of LSH is to project the data into a low-dimensional Hamming space such that each hash function $h_m(\boldsymbol{x}_i)$ for $m = 1, \ldots, M$, satisfies the *local sensitivity hashing property*:

$$P\left[h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)\right] = \mathrm{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

where $P\left[h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)\right]$ is the *probability of collision* and $\mathrm{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ represent the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

LSH is a data-independent method and the hash functions $h_m(\cdot)$ (for $m = 1, \ldots, M$) consist of random projections followed by rounding:

$$h_m(\boldsymbol{x}_i) = \mathrm{sgn}(\boldsymbol{w}_m^\top \boldsymbol{x}_i + b_m), \tag{1}$$

where $\mathrm{sgn}(z)$ is sign function which equal 1 for $z \geq 0$ and otherwise -1, $\boldsymbol{w}_m \in \mathbb{R}^D$ is random weight vector, each entry of which is drawn from p-stable distribution (including Gaussian distribution) [11], [12] and $b_m = -\frac{1}{N}\sum_{i=1}^N \boldsymbol{w}_m^\top \boldsymbol{x}_i$ is bias which is zero for centered data. Defining $h(\boldsymbol{x}_i) = [h_1(\boldsymbol{x}_i), \ldots, h_M(\boldsymbol{x}_i)] \in \mathbb{R}^M$, and then for random weight vector $\boldsymbol{w}_m$, the probability of collision was proven to be

$$P\left[h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)\right] \propto \left[1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\boldsymbol{x}_i^\top \boldsymbol{x}_j}{\|\boldsymbol{x}_i\|\|\boldsymbol{x}_j\|}\right)\right]^M,$$

where $\|\cdot\|$ is Euclidean norm. In practice, LSH requires multiple hash tables with long binary codes. The large value of M decreases the collision probability.

B. Spectral Hashing

Spectral hashing [7] seeks similarity-preserving binary codes $\{\boldsymbol{y}_i\}_{i=1}^N$ from a set of data points $\{\boldsymbol{x}_i\}$. Spectral hashing requires the average Hamming distance between similar neighbors to be minimized. In addition, the codes are also required to be balanced and uncorrelated. Thus, spectral hashing involves the following optimization:

$$\begin{aligned}
\arg\min_{\boldsymbol{y}} \quad & \sum_{i=1}^N \sum_{j=1}^N \mathrm{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j)\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2, \\
\text{subject to} \quad & \boldsymbol{y}_i \in \{+1, -1\}^M, \\
& \sum_{i=1}^N \boldsymbol{y}_i = 0, \\
& \frac{1}{N}\sum_{i=1}^N \boldsymbol{y}_i \boldsymbol{y}_i^\top = \boldsymbol{I}_M,
\end{aligned}$$

where $\boldsymbol{I}_M \in \mathbb{R}^{M \times M}$ is the identity matrix. The last two constraints represent the balancing condition and pairwise decorrelation condition.

The formulation in spectral hashing is equivalent to a particular form of graph partitioning, which is known to NP-hard. The problem is relaxed by discarding binary constraints $\boldsymbol{y}_i \in \{+1, -1\}^M$. Then rounding a subset of eigenvectors of the graph Laplacian of the similarity graph leads to binary codes for spectral hashing. For out of sample extension, data are assumed to be generated from separable multi-dimensional uniform distribution and eigenfunctions of the weighted Laplace-Beltrami operators defined on manifold are used to determine binary codes of unseen data points.

C. Semi-Supervised Hashing

Recently, semi-supervised hashing (SSH) [10] was developed, where both labeled and unlabeled data are used to learn binary codes. Two categories of label information are used: (1) $\mathcal{M}$ is a set of neighbor-pair in which $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{M}$ implies that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are either neighbors in a metric space or share common labels; (2) $\mathcal{C}$ is a set of nonneighbor-pair if $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{C}$ means that two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are far away in metric space or have different class labels. Given a data matrix $\boldsymbol{X} \in \mathbb{R}^{D\times N}$, one learns M hash functions yielding M-bit Hamming embedding $\boldsymbol{Y} \in \mathbb{R}^{M\times N}$. As in LSH, the mth hash function is defined as

$$h_m(\boldsymbol{x}_i) = \text{sgn}(\boldsymbol{w}_m^\top \boldsymbol{x}_i + b_m),$$

where $b_m = -\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_m^\top \boldsymbol{x}_i$ is the mean of the projected data, i.e., which is zero for centered data.

In contrast to LSH, weight vectors $\boldsymbol{w}_m$ are determined by maximizing the empirical accuracy on the labeled data for a family of hash functions. The objective function to be maximized in SSH is given by

$$\mathcal{J}_{SSH} = \sum_{m=1}^{M} \left\{ \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{M}} h_m(\boldsymbol{x}_i)h_m(\boldsymbol{x}_j) - \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{C}} h_m(\boldsymbol{x}_i)h_m(\boldsymbol{x}_j) \right\}.$$

As in spectral hashing, the balancing and pairwise decorrelation conditions are also considered. Introducing an indicator matrix $\boldsymbol{\Gamma}$, the (i,j)-entry of which is given by

$$\Gamma_{ij} = \begin{cases} 1 & \text{if } (\boldsymbol{x}_i,\boldsymbol{x}_j) \in \mathcal{M}, \\ -1 & \text{if } (\boldsymbol{x}_i,\boldsymbol{x}_j) \in \mathcal{C}, \\ 0 & \text{otherwise.} \end{cases}$$

With this indicator matrix and the following notation,

$$\begin{aligned} h(\boldsymbol{x}_i) &= [h_1(\boldsymbol{x}_i), \ldots, h_M(\boldsymbol{x}_i)] \in \mathbb{R}^{M}, \\ H(\boldsymbol{X}_l) &= [h(\boldsymbol{x}_1), \ldots, h(\boldsymbol{x}_{N_l})] \in \mathbb{R}^{M\times N_l}, \\ \boldsymbol{W} &= [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M] \in \mathbb{R}^{D\times M}. \end{aligned}$$

the optimization problem for SSH is written as:

$$\underset{H}{\arg\max} \quad \frac{1}{2}\text{tr}\left\{H(\boldsymbol{X}_l)\boldsymbol{\Gamma}H(\boldsymbol{X}_l)^\top\right\}, \tag{2}$$

$$\text{subject to} \quad \sum_{i=1}^{N} h_m(\boldsymbol{x}_i) = 0, \text{ for } m = 1, \ldots, M, \tag{3}$$

$$\frac{1}{N}H(\boldsymbol{X})H(\boldsymbol{X})^\top = \boldsymbol{I}_M. \tag{4}$$

Without loss of generality, data are centered and $\|\boldsymbol{w}_m\| = 1$. A few relaxations are used in SSH. The sign of projection is replaced by its signed magnitude, i.e., $H(\boldsymbol{X}_l) = \text{sgn}\left(\boldsymbol{W}^\top\boldsymbol{X}_l\right)$ is replaced by $\boldsymbol{W}^\top\boldsymbol{X}_l$. The balancing constraint (3) is replaced by a soft constraint involving the variance maximization. Thus, the leading eigenvectors of $\boldsymbol{X}_l\boldsymbol{\Gamma}\boldsymbol{X}_l^\top + \eta\boldsymbol{X}\boldsymbol{X}^\top$ yields the embedding solution, where η is a trade-off parameter. See [10] for more details.

III. Semi-Supervised Discriminant Hashing

We present our main contribution, which is semi-supervised discriminant hashing (SSDH). The rationale behind SSDH is to maximize the separability between binary codes learned in semi-supervised fashion. Especially, in the case of short binary codes, it is desirable to maximize the separability between them so that the performance in CBIR increases. Thus we base our SSDH on linear discriminant analysis, where hash functions are learned such that labeled data are used to maximize the separability between binary codes associated with different classes while unlabeled data are used for regularization as well as for balancing condition and pairwise decorrelation of bits.

Suppose that labeled data points $\boldsymbol{x}_i^{(k)}$ for $i = 1, \ldots, N_l$, belong to one of K classes, denoted by Ω_k for $k = 1, \ldots, K$. Denote by $|\Omega_k|$ the size of class k (the number of samples in class Ω_k). We assume that labeled data points are centered and unlabeled data points are also centered, i.e., $\boldsymbol{X}_l\boldsymbol{1}_{N_l} = 0$ and $\boldsymbol{X}_u\boldsymbol{1}_{N_u} = 0$, leading to $\boldsymbol{X}\boldsymbol{1}_N = 0$, where $\boldsymbol{1}_N$ is the N-dimensional vector of ones and $\boldsymbol{X} = [\boldsymbol{X}_l, \boldsymbol{X}_u]$. Without loss of generality, we assume that labeled data points are sorted according to their class labels, i.e.,

$$\boldsymbol{X}_l = \left[\boldsymbol{X}_l^{(1)}, \ldots, \boldsymbol{X}_l^{(K)}\right],$$

where $\boldsymbol{X}_l^{(k)} \in \mathbb{R}^{D\times|\Omega_k|}$ contains all labeled examples in class k.

In the case where data are centered, the between-class scatter matrix is written as

$$\begin{aligned} \boldsymbol{S}_B &= \sum_{k=1}^{K} |\Omega_k| \left(\frac{1}{|\Omega_k|}\sum_{j\in\Omega_k} h(\boldsymbol{x}_j)\right)\left(\frac{1}{|\Omega_k|}\sum_{j\in\Omega_k} h(\boldsymbol{x}_j)\right)^\top, \\ &= \sum_{k=1}^{K} \frac{1}{|\Omega_k|}\left(\left[H\left(\boldsymbol{X}_l^{(k)}\right)\boldsymbol{1}_{|\Omega_k|}\right]\left[H\left(\boldsymbol{X}_l^{(k)}\right)\boldsymbol{1}_{|\Omega_k|}\right]^\top\right), \\ &= \sum_{k=1}^{K} H\left(\boldsymbol{X}_l^{(k)}\right)\boldsymbol{\Pi}^{(k)}H\left(\boldsymbol{X}_l^{(k)}\right)^\top, \end{aligned} \tag{5}$$

where $H\left(\boldsymbol{X}_l^{(k)}\right) = \text{sgn}\left(\boldsymbol{W}^\top\boldsymbol{X}_l^{(k)}\right)$ and

$$\boldsymbol{\Pi}^{(k)} \in \mathbb{R}^{|\Omega_k|\times|\Omega_k|} = \frac{1}{|\Omega_k|}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}. \tag{6}$$

Thus, in compact form, the between-class scatter matrix $\boldsymbol{S}_B$ is written as

$$\boldsymbol{S}_B = H\left(\boldsymbol{X}_l\right)\boldsymbol{\Pi}\,H\left(\boldsymbol{X}_l\right)^\top, \tag{7}$$

where

$$\boldsymbol{\Pi} = \begin{bmatrix} \boldsymbol{\Pi}^{(1)} & 0 & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Pi}^{(2)} & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \boldsymbol{\Pi}^{(K)} \end{bmatrix}. \tag{8}$$

The total scatter matrix is given by

$$\boldsymbol{S}_T = H\left(\boldsymbol{X}_l\right)H\left(\boldsymbol{X}_l\right)^\top. \tag{9}$$

Therefore, Fisher's discriminant analysis leads to the following optimization:

$$\begin{aligned}\underset{H}{\arg\max} \quad & \frac{\mathrm{tr}\left\{H\left(\boldsymbol{X}_l\right)\boldsymbol{\Pi}\, H\left(\boldsymbol{X}_l\right)^\top\right\}}{\mathrm{tr}\left\{H\left(\boldsymbol{X}_l\right) H\left(\boldsymbol{X}_l\right)^\top\right\}} \\ \text{subject to} \quad & \sum_{i=1}^{N} h_m(\boldsymbol{x}_i) = 0, \ \text{for } m = 1, \ldots, M, \\ & \frac{1}{N} H(\boldsymbol{X}) H(\boldsymbol{X})^\top = \boldsymbol{I}_M, \end{aligned} \tag{10}$$

where balancing and decorrelation constraints are also placed, as in spectral hashing and SSH. Then, as in SSH, we relax the sign of projection by its signed magnitude, leading to $H(\boldsymbol{X}_l) \approx \boldsymbol{W}^\top \boldsymbol{X}_l$. This relaxation leads the balancing constraint to be satisfied if data are centered. When the number of labeled examples is not sufficient, Fisher's discriminant analysis might suffer from overfitting and numerical instability. With this relaxation, we consider the regularized discriminant analysis [13], making use of unlabeled examples for regularization, which leads to the objective function

$$\mathcal{J}_{SSDH} = \frac{\mathrm{tr}\left\{\boldsymbol{W}^\top \boldsymbol{X}_l\, \boldsymbol{\Pi}\, \boldsymbol{X}_l^\top \boldsymbol{W}\right\}}{\mathrm{tr}\left\{\boldsymbol{W}^\top \boldsymbol{X}_l \boldsymbol{X}_l^\top \boldsymbol{W} + \beta \boldsymbol{W}^\top \boldsymbol{X}_u \boldsymbol{X}_u^\top \boldsymbol{W}\right\}}, \tag{11}$$

to be maximized with respect to $\boldsymbol{W}$. We choose $\beta = 1$, leading to

$$\mathcal{J}_{SSDH} = \frac{\mathrm{tr}\left\{\boldsymbol{W}^\top \boldsymbol{X}_l\, \boldsymbol{\Pi}\, \boldsymbol{X}_l^\top \boldsymbol{W}\right\}}{\mathrm{tr}\left\{\boldsymbol{W}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{W}\right\}}. \tag{12}$$

Note that data centering eliminates the balancing constraint. Note also that due to the regularization with unlabeled examples, we do not need to impose the decorrelation constraint, since the maximization of (12) is equivalent to the following optimization:

$$\begin{aligned}\underset{\boldsymbol{W}}{\arg\max} \quad & \mathrm{tr}\left\{\boldsymbol{W}^\top \boldsymbol{X}_l\, \boldsymbol{\Pi}\, \boldsymbol{X}_l^\top \boldsymbol{W}\right\} \\ \text{subject to} \quad & \frac{1}{N}\boldsymbol{W}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{W} = \boldsymbol{I}_M. \end{aligned}$$

The optimal solution $\boldsymbol{W}$ is given by the M leading generalized eigenvectors determined by the following generalized eigenvalue problem:

$$\left[\boldsymbol{X}_l\, \boldsymbol{\Pi}\, \boldsymbol{X}_l^\top\right] \boldsymbol{W} = \left[\boldsymbol{X}\boldsymbol{X}^\top\right] \boldsymbol{W}\boldsymbol{\Lambda}. \tag{13}$$

Given a test data point $\boldsymbol{x}_*$, the corresponding M-bit binary code $\boldsymbol{y}_*$ is given by

$$[\boldsymbol{y}_*]_m = \frac{1}{2}\left\{1 + \mathrm{sgn}(\boldsymbol{w}_m^\top \boldsymbol{x}_*)\right\},$$

for $m = 1, \ldots, M$. The algorithm for SSDH is outlined in Algorithm 1.

Note that the rank of the between-class scatter matrix is bounded by $K - 1$, i.e., rank$(\boldsymbol{S}_B) \leq K - 1$, implying that the code length M is limited by $K - 1$. Thus, we modify the class indicator matrix $\boldsymbol{\Pi}$ such that

$$[\widetilde{\boldsymbol{\Pi}}^{(k)}]_{ij} = \exp\left\{\frac{1}{\sigma^2}\left\|\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)}\right\|^2\right\}, \tag{14}$$

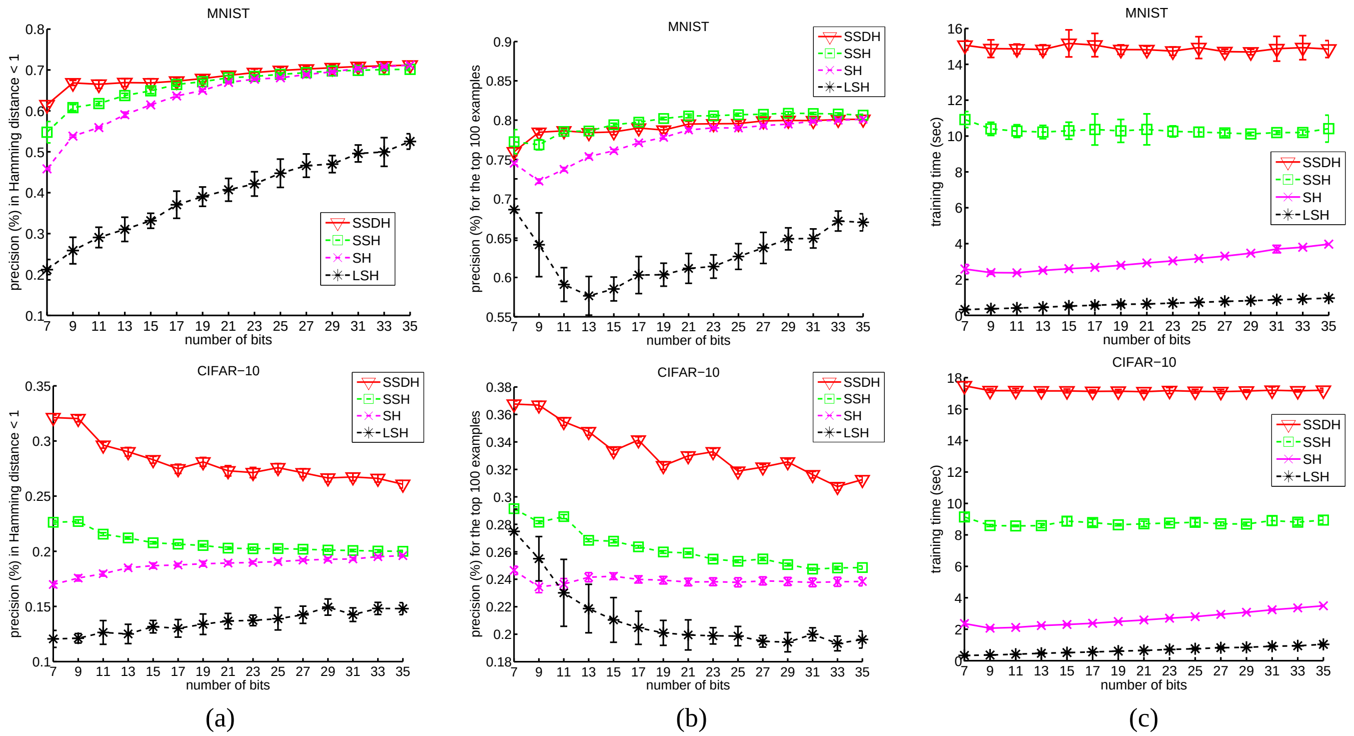
for $k = 1, \ldots, K$. In other words, each entry $\frac{1}{|\Omega_k|}$ in $\boldsymbol{\Pi}^{(k)}$ is replaced by the pairwise similarity between two points in the same class. This soft class indicator matrix improves the robustness to outliers and alleviates the rank-deficiency of the between-class scatter matrix when we choose $M \geq K$.

Algorithm 1 Semi-Supervised Discriminant Hashing

```
Input: a set of labeled and unlabeled training data training
    data $\boldsymbol{X} = [\boldsymbol{X}_l, \boldsymbol{X}_u]$, soft class indicator matrix $\widetilde{\boldsymbol{\Pi}}$ with
    the hyperparameter $\sigma$, test data point $\boldsymbol{x}_*$, and $M$ (the
    number of hash functions)
Output: binary code $\boldsymbol{y}_*$ associated with $\boldsymbol{x}_*$
1: Solve the generalized eigenvalue decomposition:
   $\left[\boldsymbol{X}_l \widetilde{\boldsymbol{\Pi}} \boldsymbol{X}_l^\top\right] \boldsymbol{W} = \left[\boldsymbol{X}\boldsymbol{X}^\top\right] \boldsymbol{W}\boldsymbol{\Lambda}$ to determine $M$
   leading eigenvectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M$.
2: Return $M$-bit binary code $\boldsymbol{y}_*$, the $i$th element of which
   is computed by
   $[\boldsymbol{y}_*]_i = \frac{1}{2}\left\{1 + \mathrm{sgn}(\boldsymbol{w}_i^\top \boldsymbol{x}_*)\right\}, \quad i = 1, \ldots, M.$
```

IV. Numerical Experiments

We compare the performance of four different methods on two benchmark datasets (MNIST and CIFAR-10), including our semi-supervised discriminant hashing (SSDH) and three existing methods (LSH [3], SH [7], SSH [10]). Two image datasets were used in our experiments: (a) **MNIST** [14] is a hand-written digit dataset, in which each image is associated with a label from 0 to 9. As MNIST consists of 70K examples, we randomly chose 60K examples as training data and the other 10K examples as test queries (We repeat this procedure ten times). For simplicity, we used the raw data (28-by-28 pixels), converting into a 784-dimensional vector; (b) **CIFAR-10** [15] is the subset of 80-million tiny image dataset. CIFAR-10 is composed of 10 different classes, and the total number of examples in CIFAR-10 is 60K. We randomly selected 50K for training data and the other was reserved for test queries, repeating this procedure ten times. To extract features for CIFAR-10, we used the three heterogenous visual features: GIST [5], Bag-of-Words, and HOG [16]. We concatenated the three visual features to construct a single vector. In this experiment, we followed the same procedure to extract GIST descriptor [6]. We obtained Gabor-filtered images for different 8 orientations and 4 scales. Each filtered image is averaged over 4-by-4 grid, which results in a $8 \times 4 \times 16 = 512$-dimensional vector



(a) (b) (c)

Figure 1. Comparison of performance and training time of different binary codes determined by our methods (SSDH) and existing methods (SSH, SH, LSH), on MNIST and CIFAR-10: (a) precision (%) in Hamming distance < 1; (b) precision (%) for the top 100 examples; (c) training time (seconds).

for a single image. For Bag-of-Words, we used the dense SIFT [17] for a local descriptor and performed k-means (k=200) on the random subset of training data to construct the visual codewords, resulting in a 200-dimensional vector for a single image. Finally, for HOG descriptor, we used 3-by-3 cells and 9 bins of orientation histogram, constructing a 81-dimensional vector for a single image. Therefore, we used $512 + 200 + 81 = 793$-dimensional vector for a single image. Without loss of generality, we assume that the data are centered and the feature values for each dimension are normalized into $[0, 1]$.

Since MNIST and CIFAR-10 provide the label information for every example, we used 20% examples for labeled data and the rest of them for unlabeled data in cases of SSDH and SSH. For the soft indicator matrix in SSDH, we used the Gaussian kernel $\exp(-\frac{1}{\sigma^2}||\boldsymbol{x} - \boldsymbol{y}||_2^2)$, where σ is a kernel width. To set the kernel width (σ), we used the 3-fold cross validation on training data with the following range: $\sigma \in \{0.5\beta, \beta, \ldots, 3.5\beta, 4.0\beta\}$, where β is the average pairwise distance for the sampled data. In this experiment, we sampled 100 data points to compute β. For LSH, we used zero mean and identity covariance matrix to construct the random weight vector. For SH, we used the online-available code from the authors [7]

Fig. 1 shows the comparison of the three representative hashing methods (SSH, SH, LSH) and the proposed methods (SSDH). For quantitative evaluations, we were based on the precision of a query: the ratio of semantically related examples among the retrieved ones. Since our test datasets are fully labeled, we can consider that the semantically related examples share the same label. Specifically, we first computed the precision of a query within Hamming distance 1 (threshold). We increased the threshold until the number of retrieved examples is at least 100. Second, we computed the precision for the top 100 retrieved examples of a query. To do that, we first retrieved more than 100 candidate examples according to Hamming distance, then re-sorted the candidate examples according to the original descriptor.

In Fig. 1, the first row represents the results for MNIST, and the second row does the result for CIFAR-10. All experiments are conducted varying the number of bits. The first column (a) is the results for the precision (%) within Hamming distance 1, the second column (b) is for the precision (%) for the top 100 examples and the third column (c) represents the training time (sec) for hash functions.

Fig. 1 (a) and (b) show that the precision of SSDH is reported higher than the previous hashing methods, which indicates that SSDH can produce the more discriminant binary codes. Especially, in case of short binary codes, we observe that the superiority of SSDH is exaggerated. According to Fig. 1 (c), we can observe that SSDH requires reasonable training time, in spite of computing the soft indicator matrix and the generalized eigenvalue decomposition.

Fig. 2 represents the qualitative results with 9 bits for

CIFAR-10. We retrieved the candidate images whose Hamming distance between a query is less than 1, and re-sorted the retrieved images according to their original descriptors. In Fig. 2, SSDH outperforms to retrieve the semantically related images for a test query, compared to the existing hashing methods. For examples, SSH, SH, and LSH retrieve unrelated images (airplane or car) marked by a red triangle.

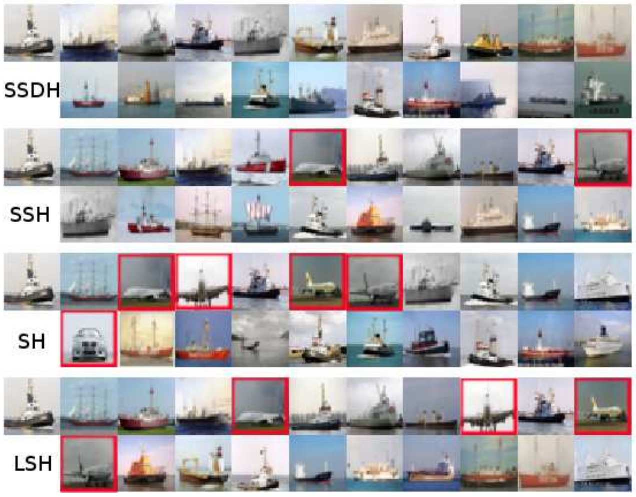


Figure 2. Qualitative Results on CIFAR-10 for four different hashing methods (SSDH, SSH, SH and LSH) using 9 bits. The leftmost image is the query image and twenty nearest images are displayed for each method.

V. Conclusions

We have presented a method for semi-supervised hashing based on Fisher discriminant analysis such that labeled data are used to maximize the separability between binary codes in different classes while unlabeled data are used for regularization. We demonstrated the high performance of SSDH on two image datasets, MNIST and CIFAR-10, in the task of image retrieval. We emphasize that SSDH outperforms existing methods such as LSH, SH, and SSH, especially in the case of short binary codes.

Acknowledgments: This work was supported by NIPA-MSRA Creative IT/SW Research Project, NIPA ITRC Support Program (NIPA-2011-C1090-1131-0009), MEST Converging Research Center Program (No. 2011K000673), and NRF World Class University Program (R31-10100).

References

[1] J. H. Friedman, J. L. Bentley, and R. A. Finkel, “An algorithm for finding best matches in logarithmic expected time,” *ACM Transactions on Mathematical Softwares*, vol. 3, no. 3, pp. 209–226, 1977.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An optimal algorithm for approximate nearest neighbor searching,” *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[3] A. Gionis, P. Indyk, and R. Motawani, “Similarity search in high dimensions via hashing,” in *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1999.

[4] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[6] A. Torralba, R. Fergus, and Y. Weiss, “Small codes and large image databases for recognition,” in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, 2008.

[7] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *Advances in Neural Information Processing Systems (NIPS)*, vol. 20. MIT Press, 2008.

[8] R. Salakhutdinov and G. Hinton, “Semantic hashing,” in *Proceeding of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[9] G. Shakhnarovich, P. Viola, and T. Darrel, “Face pose estimation with parameter sensitive hashing,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, Beijing, China, 2003.

[10] J. Wang, S. Kumar, and S. F. Chang, “Semi-supervised hashing for scalable image retrieval,” in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, 2010.

[11] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, “Locality sensitive hashing scheme based on p-stable distributions,” in *Proceedings of the Annual ACM Symposium on Computational Geometry (SoCG)*, 2004.

[12] M. S. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, 2002.

[13] J. H. Friedman, “Regularized discriminant analysis,” *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 165–175, 1989.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] A. Krizhevsky and G. E. Hinton, “Learning multiple layers of features from tiny images,” Computer Science Department, University of Toronto, Tech. Rep., 2009.

[16] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, 2005.

[17] L. Fei-Fei and P. Perona, “A Bayesian hierarchical model for learning natural scene categories,” in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, 2005.