Semi-Supervised Hashing for Scalable Image Retrieval

Jun Wang
Columbia University
New York, NY, 10027
jwang@ee.columbia.edu

Sanjiv Kumar
Google Research
New York, NY, 10011
sanjivk@google.com

Shih-Fu Chang
Columbia University
New York, NY, 10027
sfchang@ee.columbia.edu

Abstract

Large scale image search has recently attracted considerable attention due to easy availability of huge amounts of data. Several hashing methods have been proposed to allow approximate but highly efficient search. Unsupervised hashing methods show good performance with metric distances but, in image search, semantic similarity is usually given in terms of labeled pairs of images. There exist supervised hashing methods that can handle such semantic similarity but they are prone to overfitting when labeled data is small or noisy. Moreover, these methods are usually very slow to train. In this work, we propose a semi-supervised hashing method that is formulated as minimizing empirical error on the labeled data while maximizing variance and independence of hash bits over the labeled and unlabeled data. The proposed method can handle both metric as well as semantic similarity. The experimental results on two large datasets (up to one million samples) demonstrate its superior performance over state-of-the-art supervised and unsupervised methods.

1. Introduction

Due to explosive growth of visual content on the Web, such as personal photographs and videos, there is an emerging need of searching visually relevant images and videos from very large databases. Besides the widely-used text-based commercial search engines, content based image retrieval (CBIR) has attracted substantial attention over the past decade. Instead of taking query words as input, CBIR techniques directly take an image q as query and try to return its nearest neighbors from a given database of images $\mathcal{X}$ using a pre-specified similarity metric $\mathcal{M}$. Since databases containing even billions of samples are not that uncommon, such large-scale search demands highly efficient and accurate retrieval methods.

Exhaustively comparing the query q with each sample in the database $\mathcal{X}$ is infeasible because linear complexity $\mathcal{O}(|\mathcal{X}|)$ is not scalable in practical settings. For example, the photo sharing website **Flickr** has over 4 billion images. Another visual content sharing website **YouTube** receives more than 20 hours of uploaded videos per minute. Besides the scalability issue, most large scale CBIR applications also suffer from the *curse of dimensionality* since visual descriptors usually have hundreds or even thousands of dimensions. Therefore, beyond the infeasibility of exhaustive search, storage of the original data also becomes a big problem. Instead of doing exact nearest neighbor search through linear scan, a fast and accurate indexing method with sublinear ($o(|\mathcal{X}|)$), logarithmic ($\mathcal{O}(\log|\mathcal{X}|)$) or even constant ($\mathcal{O}(1)$) query time is desired for realizing large scale CBIR applications. Over the past decade, several approximate nearest neighbor (ANN) search techniques have been developed for large scale applications. Although there exist many tree-based methods e.g., [3, 1, 11, 14, 10], for applications with memory constraints, hashing based ANN techniques have attracted more attention. They have constant query time and also substantially reduced storage as they usually store only compact binary codes for each point in $\mathcal{X}$.

Hashing methods can be divided into two main categories: unsupervised methods and supervised methods. Unsupervised methods use just the unlabeled data $\mathcal{X}$ to generate binary codes for the given points. Locality Sensitive Hashing (LSH) [4] is arguably one of the most popular unsupervised hashing methods in computer vision. Its kernelized version has also been developed in [9]. Another effective method called Spectral Hashing (SH) was proposed recently by Weiss et al. [17]. Since unsupervised methods do not require any labeled data, their parameters are easy to learn given a pre-specified distance metric. However, in vision problems, sometimes similarity (or distance) between data points is not defined with a simple metric. Metric similarity of image descriptors may not preserve semantic similarity. For example, Figure 2 shows erroneous retrieval results by LSH and SH without considering the image label information. Ideally, one would like to provide pairs of images that one believes contain 'similar' or 'dissimilar' images. From such pairwise labeled data, one would like the

978-1-4244-6985-7/10/$26.00 ©2010 IEEE

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on July 23,2020 at 03:17:34 UTC from IEEE Xplore. Restrictions apply.

Method	Projection Dependency	Learning Paradigm
LSH [4]	data-independent	unsupervised
SH [17]	data-dependent	unsupervised
RBMs [6][16]	–	unsupervised/supervised
SSH	data-dependent	semi-supervised

Table 1. The conceptual comparison of the proposed SSH method with LSH, SH and RBMs.

hashing mechanism to automatically generate codes that respect this semantic similarity.

Many supervised hashing methods have been developed to handle this issue. Modifying unsupervised LSH, in [7], authors have suggested merging LSH with a learned Mahalanobis metric to reflect semantic indexing. A Boosting Similarity Sensitive Coding (BoostSSC) technique was proposed by [13] which tries to learn a series of weighted hashing functions from labeled data. A deep neural network stacked with Restricted Boltzmann Machines (RBMs)[1] was recently applied to learn compact binary codes from high dimensional inputs [6, 16], which has shown superior performance over BoostSSC. One of the problems with all of these supervised methods is that they are much slower in comparison to the unsupervised methods. Another problem stems from limited or noisy training data. The performance of these methods degrades with less training data due to overfitting.

In this paper, we propose a *Semi-Supervised Hashing* (SSH) technique that can leverage semantic similarity using labeled data while remaining robust to overfitting. SSH is also much faster than existing supervised hashing methods and can be easily scaled to large datasets. The SSH problem is cast as a data-dependent projection learning problem. We provide a rigorous formulation in which a supervised term tries to minimize the empirical error on the labeled data while an unsupervised term provides effective regularization by maximizing desirable properties like variance and independence of individual bits. We show that the resulting formulation can be easily relaxed and solved as a standard eigenvalue problem. Furthermore, by relaxing the orthogonality constraints, one can get even better hash codes at no added computational cost. Table 1 summarizes a taxonomy of popular methods and places our method in context.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce the related work on representative hashing methods. Section 3 presents our proposed approach, i.e. *Semi-Supervised Hashing* (SSH). Section 4 provides extensive experimental validation on real image datasets. The conclusions and future work are given in Section 5.

[1]RBMs use both labeled and unlabeled data but unlabeled data is used in a pre-training phase, which provides a good initialization for the supervised back-propagation phase. Hence RBM is a supervised method.

2. Background and Related Work

Given a set containing n points, $\mathcal{X} = \{\mathbf{x}_i\}, i = 1, \cdots, n$ and $\mathbf{x}_i \in \mathbb{R}^D$, the objective in nearest neighbor search is to find a set of nearest neighbors $\mathcal{R} \subset \mathcal{X}$ for a given query q. For large-scale applications, to avoid excessive computational and memory costs, one would like to instead do an Approximate Nearest Neighbor (ANN) search with sublinear query complexity [12]. For example, the *ϵ-approximate nearest neighbor* returns sample r to query q satisfying $d(r, q) \leq (1+\epsilon) \cdot d(r^*, q)$, where r^* is the nearest neighbor point of q and $\epsilon > 0$.

In the next section, in addition to the popularly used LSH, we briefly review the state-of-the-art methods from supervised as well as unsupervised domains. Specifically, we discuss SH and RBM techniques along with their pros and cons for the application of image retrieval.

2.1. Locality Sensitive Hashing (LSH)

A key ingredient of *Locality Sensitive Hashing* is mapping similar samples to the same bucket with high probability. In other words, the property of locality in the original space will be largely preserved in the hamming space. More precisely, the hashing functions $h(\cdot)$ from LSH family satisfy the following elegant locality preserving property:

$$P\{h(\mathbf{x}) = h(\mathbf{y})\} = sim(\mathbf{x}, \mathbf{y}) \tag{1}$$

where the similarity measure can be directly linked to the distance function d, for example, $sim(\mathbf{x}, \mathbf{y}) = \exp\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\}$. A typical category of LSH functions consists of random projections and thresholds as:

$$h(x) = sign(\mathbf{w}^\top \mathbf{x} + b) \tag{2}$$

where $\mathbf{w}$ is a random hyperplane and b is a random intercept. Clearly, the random vector $\mathbf{w}$ is data-independent, which is usually constructed by sampling each component of $\mathbf{w}$ randomly from a p-stable distribution e.g., standard Gaussian [2]. Although there exists an asymptotic theoretical guarantee for random projection based LSH, it is not very efficient in practice since it requires multiple tables with long codes [4]. For example, for a normalized data set ($\|\mathbf{x}\| = 1$) with zero mean, the approximately balanced partition is obtained with $b = 0$. Constructing a total of l K-bit length hash tables $H(x) = [h_1(x), \cdots, h_K(x)]$ provides the following collision probability:

$$P\{H(\mathbf{x}) = H(\mathbf{y})\} = l \cdot \left[1 - \frac{cos^{-1}\mathbf{x}^\top \mathbf{y}}{\pi}\right]^K \tag{3}$$

For a large scale application, the value of K should be considerably large to reduce the size of each hash bucket (i.e., the number of samples falling in the same bucket). However

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on July 23,2020 at 03:17:34 UTC from IEEE Xplore. Restrictions apply.

a large value of K decreases the collision probability between similar samples. In order to overcome this drawback, multiple hash tables have to be constructed. Obviously, this is inefficient due to extra storage cost and larger query time.

2.2. Spectral Hashing (SH)

Due to the limitation of random projection based LSH approach, machine learning techniques have been applied to improve the efficiency of hashing. Particularly, *Spectral Hashing* (SH) was recently proposed to design compact binary codes for ANN search. Besides the common property of maintaining sample similarity in the reduced hamming space, SH requires the codes to be balanced and uncorrelated. Strictly speaking, SH codes $H(\mathbf{x}) = \{h_k(\mathbf{x})\}, k = 1, \cdots, K$ satisfy the following criteria [17]:

$$\min \sum_{ij} sim(\mathbf{x}_i, \mathbf{x}_j) \|H(\mathbf{x}_i) - H(\mathbf{x}_j)\|^2 \tag{4}$$

$$\text{subject to:} \quad h_k(\mathbf{x}_i) \in \{-1, 1\}$$

$$\sum_i h_k(\mathbf{x}_i) = 0, k = 1, \cdots, K$$

$$\sum_i h_k(\mathbf{x}_i) h_l(\mathbf{x}_i) = 0, \text{for } \; k \neq l$$

The direct solution for the above optimization is non-trivial since even a single bit partition (K=1) is a balanced graph partition problem, which is NP hard. The combination of K-bit balanced partition will be even harder because of the pairwise independence constraints. After relaxing the constraints, the above optimization was solved using spectral graph analysis. Especially, with the assumption of uniform data distribution, the spectral solution can be efficiently generalized to out of samples extension with a close form solution [17].

The final SH algorithm consists of three key steps: 1) extraction of maximum variance directions through Principal Component Analysis (PCA) on the data; 2) direction selection, which prefers to partition projected dimensions with large range and small spatial frequency; 3) partition of projected data by a sinusoidal function with previously computed angular frequency. SH has been shown to be very effective in encoding large-scale, low-dimensional data since the important PCA directions are selected multiple times to create binary bits. However, for high dimensional problems ($D >> K$) where many directions contain enough variance, usually each PCA direction is picked only once. This is because the top few projections have similar range and thus, a low spatial frequency ($k = 1$) is preferred. In this case, SH approximately replicates a PCA projection followed by a mean partition. In SH, the projection directions are data dependent but learned in an unsupervised manner. Moreover, the assumption of uniform data distribution is usually not true for real-world data.

2.3. Restricted Boltzmann Machines (RBMs)

Learning deep belief networks via stacking *Restricted Boltzmann Machines* (RBMs) to obtain compact binary codes was recently proposed in [6]. This type of trained networks are capable of capturing higher order correlations between different layers of the network. Since the network structure gradually reduces the number of units in each layer, the high-dimensional input can be projected to a much more compact binary vector space.

A practical implementation of RBMs has two critical stages: unsupervised pre-training and supervised fine-tuning. The greedy pre-training phase is progressively executed layer by layer from input to output. After achieving convergence of the parameters of a layer via contrastive divergence, the derived activation probabilities are fixed and treated as input to drive the training of the next layer. During the fine-tuning stage, the labeled data is used to help refine the trained network through back-propagation. Specifically, a cost function is first defined to estimate the number of correctly classified points in the training set [5]. Then, the network weights are refined to maximize this objective function through gradient descent. RBM-based binary encoding involves estimating a large number of weights. For example, the RBMs structure used in [16] has five layers of size $512 - 512 - 256 - 32$ nodes requiring a total of 663552 weights to learn. This not only involves an extremely costly training procedure, but also demands sufficient training data for fine-tuning.

3. Semi-Supervised Hashing

In this section, we present our hashing method, i.e. *Semi-Supervised Hashing* (SSH). In the setting of SSH, one is given a set of n points, $\mathcal{X} = \{\mathbf{x}_i\}, \; i = 1 \ldots n, \mathbf{x}_i \in \mathbb{R}^D$, in which a fraction of pairs are associated with two categories of label information, $\mathcal{M}$ and $\mathcal{C}$. Specifically, a pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$ is denoted as a neighbor-pair in which $\mathbf{x}_i$ and $\mathbf{x}_j$ are either neighbors in a metric space or share common class labels. Similarly, $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$ is called a nonneighbor-pair if two samples are far away in metric space or have different class labels. Let us denote the data matrix by $\mathbf{X} \in \mathbb{R}^{D \times n}$ where each column is a data point. Also, suppose there are L points, $L < n$, which are associated with at least one of the categories $\mathcal{M}$ or $\mathcal{C}$. Let us denote the matrix formed by these L columns of $\mathbf{X}$ as $\mathbf{X}_l \in \mathbb{R}^{D \times L}$. The goal of SSH is to learn hash functions that minimize the error on the labeled training data $\mathbf{X}_l$, while maximally satisfying the desirable properties of hashing e.g., independence of bits and balanced partitioning. We start the discussion of our learning paradigm with the basic formulation of SSH.

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on July 23,2020 at 03:17:34 UTC from IEEE Xplore. Restrictions apply.

3.1. Formulation

Hashing aims to map the data $\mathbf{X} \in \mathbb{R}^{D\times n}$ to a Hamming space to obtain its compact representation. Suppose we want to learn K hash functions leading to a K-bit Hamming embedding of $\mathbf{X}$ given by $\mathbf{Y} \in \mathbb{B}^{K\times n}$. Without loss of generality, let $\mathbf{X}$ be normalized to have zero mean. In this work, we use linear projection coupled with mean thresholding as a hash function. In other words, given a vector $\mathbf{w}_k \in \mathbb{R}^D$, the k^{th} hash function is defined as,

$$h_k(\mathbf{x}_i) = \text{sgn}(\mathbf{w}_k^\top \mathbf{x}_i + b_k) \tag{5}$$

where b_k is the mean of the projected data, i.e., $b_k = -\frac{1}{n}\sum_{j=1}^{n} \mathbf{w}_k^\top \mathbf{x}_j = 0$ since $\mathbf{X}$ is zero-mean. One can get the corresponding binary bit as,

$$y_{ki} = \frac{1}{2}(1 + h_k(\mathbf{x}_i)) = \frac{1}{2}(1 + \text{sgn}(\mathbf{w}_k^\top \mathbf{x}_i)) \tag{6}$$

Let $\mathbf{H} = [h_1, \ldots, h_K]$ be a sequence of K hash functions and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_K] \in \mathbb{R}^{D\times K}$. We want to learn a $\mathbf{W}$ that gives the same bits for $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$ and different bits for $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$. We define the following objective function measuring the empirical accuracy on the labeled data for a family of hash functions $\mathbf{H}$:

$$J(\mathbf{H}) = \sum_k \left\{ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} h_k(\mathbf{x}_i)h_k(\mathbf{x}_j) - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} h_k(\mathbf{x}_i)h_k(\mathbf{x}_j) \right\} \tag{7}$$

One can express the above objective function in a compact matrix form by first defining a matrix $\mathbf{S} \in \mathbb{R}^{L\times L}$ incorporating the pairwise labeled information from $\mathbf{X}_l$ as:

$$S_{ij} = \begin{cases} 1 & : \ (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ -1 & : \ (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ 0 & : \ otherwise. \end{cases} \tag{8}$$

Also, suppose $\mathbf{H}(\mathbf{X}_l) \in \mathbb{R}^{K\times L}$ maps the points in $\mathbf{X}_l$ to their K-bit hash codes. Then, the objective function $J(\mathbf{H})$ can be represented as,

$$J(\mathbf{H}) = \frac{1}{2}\mathbf{tr}\left\{\mathbf{H}(\mathbf{X}_l)\ \mathbf{S}\ \mathbf{H}(\mathbf{X}_l)^\top\right\} \tag{9}$$
$$\Rightarrow J(\mathbf{W}) = \frac{1}{2}\mathbf{tr}\left\{\text{sgn}(\mathbf{W}^\top\mathbf{X}_l)\ \mathbf{S}\ \text{sgn}(\mathbf{W}^\top\mathbf{X}_l)^\top\right\}$$

where $\text{sgn}(\mathbf{W}^\top\mathbf{X}_l)$ is the matrix of signs of individual elements. Since the above function measures only the empirical accuracy, it is prone to overfitting especially when the size of labeled set is small compared to the entire dataset (i.e. $L \ll n$). To get better generalization ability, one needs to add "regularization" by incorporating conditions that lead to desirable properties of hash codes, independent of the performance on the labeled training data. These additional regularization terms use all the data $\mathbf{X}$ including the unlabeled ones leading to a semi-supervised learning paradigm.

Motivated by spectral hashing [17], we would like to generate hash codes in which bits are independent and each bit maximizes the information by generating a balanced partition of the data. Further, we relax the stronger condition of independence to pairwise decorrelation of bits. The balancing property specifies that each hash function $h_k(\cdot)$ should partition the entire dataset $\mathbf{X}$ into two sets of equal size. In summary, we intend to learn optimal hash functions $\mathbf{H}^*$ by maximizing the modified objective function with constraints as:

$$\begin{aligned} \mathbf{H}^* \quad &= \quad \arg\max_{\mathbf{H}} J(\mathbf{H}) \\ \text{subject to} \quad &\sum_{i=1}^{n} h_k(\mathbf{x}_i) = 0, k = 1, \ldots, K \\ &\frac{1}{n}\mathbf{H}(\mathbf{X})\mathbf{H}(\mathbf{X})^\top = \mathbf{I} \end{aligned} \tag{10}$$

The above problem is difficult to solve even without the constraints since the objective function $J(\mathbf{H})$ itself is non-differentiable. Furthermore, the balancing contraint makes the problem NP hard [17]. In the next section, we propose to relax the objective function as well as the constraints to obtain an approximate solution.

3.2. Relaxing Objective Function

In the relaxed version of the objective function, we replace the sign of projection with its *signed magnitude* in (7). This relaxation is quite intuitive in the sense that it not only desires similar points to have the same sign but also large projection magnitudes, meanwhile projecting dissimilar points not only with different signs but also as far as possible. With this relaxation, the new objective can be directly written as a function of $\mathbf{W}$ as,

$$J(\mathbf{W}) = \sum_k \left\{ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} \mathbf{w}_k^\top \mathbf{x}_i \mathbf{x}_j^\top \mathbf{w}_k - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} \mathbf{w}_k^\top \mathbf{x}_i \mathbf{x}_j^\top \mathbf{w}_k \right\} \tag{11}$$

Without loss of generality, we also assume $\|\mathbf{w}_k\| = 1, \ \forall k$. The above function can be expressed in a matrix form as,

$$J(\mathbf{W}) = \frac{1}{2}\mathbf{tr}\left\{\mathbf{W}^\top\mathbf{X}_l\mathbf{S}\mathbf{X}_l^\top\mathbf{W}\right\}. \tag{12}$$

We now address how to relax the balancing constraint $\sum_i h_k(\mathbf{x}_i) = 0$ by first showing that this constraint is equivalent to maximizing the variance for the k^{th} bit.

Proposition 3.1 (maximum variance condition)**.** *A hash function with maximum variance on data* $\mathbf{X}$ *must satisfy the balancing constraint, and vice-versa.*

$$\sum_i h(\mathbf{x}_i) = 0 \iff \max var[h(\mathbf{x})]$$

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on July 23,2020 at 03:17:34 UTC from IEEE Xplore. Restrictions apply.

Proof. In the set $\mathbf{X}$, suppose m points are assigned a hash value -1, and the remaining $n-m$ points are associated with a hash value 1. Then, the expected hash value is,

$$\mu = E[h(\mathbf{x})] = \frac{1}{n}\sum_i h(\mathbf{x}_i) = \frac{n-2m}{n}$$

and the variance is:

$$\begin{aligned} &var[h(\mathbf{x})] = E[(h(\mathbf{x})-\mu)^2] \\ &= (1-\frac{n-2m}{n})^2\frac{(n-m)}{n} + (-1-\frac{n-2m}{n})^2\frac{m}{n} \\ &= \frac{4}{n^2}(mn-m^2) \end{aligned}$$

Clearly, $var[h(\mathbf{x})]$ is concave with respect to m and its maximum is reached at $m = n/2$, i.e. $\mu = 0$. Also since $var[h(\mathbf{x})]$ has a unique maximum, it is easy to see that the balanced partitioning also maximizes the variance of hashing function. □

Since we use the sign of projections to generate bits, it is hard to ensure perfectly balanced partitions. Hence, we replace the hard balancing constraint by a "soft" constraint maximizing the variance of the bits as:

$$J(\mathbf{W}) = \frac{1}{2}\mathbf{tr}\left[\mathbf{W}^\top\mathbf{X}_l\mathbf{S}\mathbf{X}_l^\top\mathbf{W}\right] + \frac{\eta}{2}\sum_k E[\|h_k(\mathbf{x})-\mu_k\|^2]. \quad (13)$$

Here η is a positive scalar, which relatively weights the variance based regularization term. Furthermore, to avoid dealing with non-differentiable $\mathrm{sgn}(\cdot)$ function, we maximize directly the variance of the projected data[2]. With this assumption, one can rewrite (13) as:

$$J(\mathbf{W}) = \frac{1}{2}\mathbf{tr}\left[\mathbf{W}^\top\mathbf{X}_l\mathbf{S}\mathbf{X}_l^\top\mathbf{W}\right] + \frac{\eta}{2}\sum_k E[\|\mathbf{w}_k^\top\mathbf{x}\|^2], \quad (14)$$

since $E[\mathbf{w}_k^\top\mathbf{x}] = 0$. One can further express the variance term in a matrix form as,

$$\begin{aligned} &\sum_k E[\|\mathbf{w}_k^\top\mathbf{x}\|^2] = \sum_k E[\mathbf{w}_k^\top\mathbf{x}\mathbf{x}^\top\mathbf{w}_k] \\ &= \frac{1}{n}\mathbf{tr}\left[\mathbf{W}^\top\mathbf{X}\mathbf{X}^\top\mathbf{W}\right] \end{aligned} \quad (15)$$

Replacing (15) in (14), and absorbing n in η, one can rewrite the overall objective function as:

$$\begin{aligned} J(\mathbf{W}) &= \frac{1}{2}\mathbf{tr}\left[\mathbf{W}^\top\mathbf{X}_l\mathbf{S}\mathbf{X}_l^\top\mathbf{W}\right] + \frac{\eta}{2}\mathbf{tr}\left[\mathbf{W}^\top\mathbf{X}\mathbf{X}^\top\mathbf{W}\right] \\ &= \frac{1}{2}\mathbf{tr}\{\mathbf{W}^\top\mathbf{M}\mathbf{W}\} \end{aligned} \quad (16)$$

[2]If data is unit-norm, one can show that the variance of the projected data provides a lower bound on the maximum variance of bits.

where $\mathbf{M} = \mathbf{X}_l\mathbf{S}\mathbf{X}_l^\top + \eta\mathbf{X}\mathbf{X}^\top$. Next, we relax the pairwise decorrelation of bits by imposing orthogonality constraints on the projection directions, which coupled with unit-norm assumption leads to new constraints $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$. Now, the learning of optimal projections $\mathbf{W}$ becomes a typical eigenproblem, which can be easily solved by doing an eigenvalue decomposition on matrix $\mathbf{M}$:

$$\begin{aligned} \max_{\mathbf{W}} J(\mathbf{W}) &= \sum_{k=1}^{K}\lambda_k \\ \mathbf{W}^* &= [\mathbf{e}_1 \cdots \mathbf{e}_K] \end{aligned} \quad (17)$$

where $\lambda_1 > \lambda_2 > \cdots > \lambda_K$ are the top eigenvalues of $\mathbf{M}$ and $\mathbf{e}_k, k = 1, \cdots, K$ are the corresponding eigenvectors.

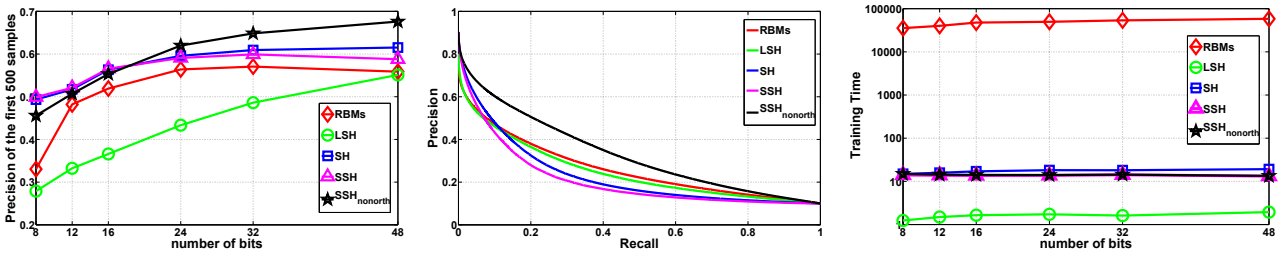
To summarize, the objective function in (16) consists of two components. The first (supervised) term is the empirical accuracy of the learned hash functions on the pairwise labeled data, while the second (unsupervised) term is a regularizer that prefers those directions that maximize the variance of the projections subject to orthogonality constraints. Mathematically it is very similar to finding maximum variance direction using PCA except that the original covariance matrix gets "adjusted" by another matrix arising from the labeled data. Hence, our framework provides an intuitive and easy way to learn hashing in a semi-supervised paradigm.

3.3. Relaxing Orthogonality Constraints

In the previous section, we imposed orthogonality constraints on the projection directions in order to approximately decorrelate the hash bits. However, these orthogonality constraints sometimes lead to a practical problem. It is well known that for most real-world datasets, most of the variance is contained in top few projections. The orthogonality constraints force one to progressively pick those directions that have very low variance, substantially reducing the quality of lower bits, and hence the whole embedding. We empirically verify this behavior in Section 4. Depending on the application, it may make sense to pick a direction that is not necessarily orthogonal to the previous directions but has higher variance as well as low empirical error on the labeled set. On the other hand, one doesn't want to pick a previous direction again since the fixed thresholds will generate the same hash codes in our case. Hence, instead of imposing hard orthogonality constraints, we convert them into a penalty term added to the objective function. This allows the learning algorithm to pick suitable directions by balancing various terms. With this, one can write the new objective function as,

$$\begin{aligned} J(\mathbf{W}) &= \frac{1}{2}\mathbf{tr}\{\mathbf{W}^\top\mathbf{M}\mathbf{W}\} - \frac{\rho}{2}\|\mathbf{W}^\top\mathbf{W}-\mathbf{I}\|_{\mathcal{F}}^2 \\ &= \frac{1}{2}\mathbf{tr}\{\mathbf{W}^\top\mathbf{M}\mathbf{W}\} - \frac{\rho}{2}\mathbf{tr}\left[(\mathbf{W}^\top\mathbf{W}-\mathbf{I})^\top(\mathbf{W}^\top\mathbf{W}-\mathbf{I})\right]. \end{aligned} \quad (18)$$

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on July 23,2020 at 03:17:34 UTC from IEEE Xplore. Restrictions apply.



(a) (b) (c)

Figure 1. Experimental results on MNIST digit dataset. a): The precision of the first 500 retrieved neighbors; b): The precision recall curve using 48-bit codes; c) Training cost (in seconds) using different number of bits.

The new formulation has certain tolerance to non-orthogonality, which is modulated by a positive coefficient ρ. However, the above objective function is non-convex and there is no easy way to find the global solution unlike the previous case. To maximize with respect to $\mathbf{W}$,

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = 0 \Rightarrow (\mathbf{W}\mathbf{W}^\top - \mathbf{I} - \frac{1}{\rho}\mathbf{M})\mathbf{W} = 0 \qquad (19)$$

Though the above equation admits an unlimited number of solutions since $\mathbf{W}$ has a non-empty nullspace, we can obtain a solution by ensuring,

$$\mathbf{W}\mathbf{W}^\top\mathbf{W} = \left(\mathbf{I} + \frac{1}{\rho}\mathbf{M}\right)\mathbf{W}. \qquad (20)$$

One can get a simple solution for the above condition if $\mathbf{I} + \frac{1}{\rho}\mathbf{M}$ is positive definite. From (16), $\mathbf{M}$ is symmetric but not necessarily positive definite. Let $\mathbf{Q} = \mathbf{I} + \frac{1}{\rho}\mathbf{M}$. Clearly, $\mathbf{Q}$ is also symmetric. In the following proposition we show that $\mathbf{Q}$ is positive definite if the coefficient ρ is chosen appropriately.

Proposition 3.2. *The matrix* $\mathbf{Q}$ *is positive definite if* $\rho > \max(0, -\bar{\lambda}_{min})$*, where* $\bar{\lambda}_{min}$ *is the smallest eigenvalue of* $\mathbf{M}$*.*

Proof. By definition in (18), $\rho > 0$. Since $\mathbf{M}$ is symmetric, it can be represented as $\mathbf{M} = \mathbf{U}diag(\lambda_1, \cdots, \lambda_D)\mathbf{U}^\top$ where all λ_i's are real. Let $\bar{\lambda}_{min} = \min(\lambda_1, \cdots, \lambda_D)$. Then $\mathbf{Q}$ can be written as

$$\begin{aligned}\mathbf{Q} &= \mathbf{I} + \mathbf{U}diag\left(\frac{\lambda_1}{\rho}, \cdots, \frac{\lambda_D}{\rho}\right)\mathbf{U}^\top \\ &= \mathbf{U}diag\left(\frac{\lambda_1}{\rho} + 1, \cdots, \frac{\lambda_D}{\rho} + 1\right)\mathbf{U}^\top\end{aligned}$$

Clearly, $\mathbf{Q}$ will have all eigenvalues positive if $\frac{\lambda_{min}}{\rho} + 1 > 0 \Rightarrow \rho > -\lambda_{min}$. □

When $\mathbf{Q}$ is positive definite, one can easily verify that the following solution of $\mathbf{W}$ satisfies Eq. (20):

$$\mathbf{W}^*_{nonorth} = \mathbf{U}_k\mathbf{\Sigma}_k^{1/2}\mathbf{U}_k^\top \qquad (21)$$

where $\mathbf{U}_k$ are the top k eigenvectors of $\mathbf{M}$ and $\mathbf{\Sigma}_k$ is a diagonal matrix consisting of entries $(1 + \lambda_i/\rho)$ where λ_i is one of the top k eigenvalues. It is interesting to note that the solution for non-orthogonal $\mathbf{W}$ is also obtained by direct eigendecomposition of $\mathbf{M}$ similar to the orthogonal version in (17) resulting in negligible computational overhead.

4. Experiments

We evaluated the two versions of our method, SSH and SSH$_{nonorth}$ (with orthogonality constraint relaxed) on the MNIST digit dataset and a Gist dataset, and compared with three state-of-the-art binary coding methods, LSH, SH, and RBMs.

4.1. MNIST Digits

The MNIST dataset consists of a total of 70000 handwritten digit samples, each of size 28×28 pixels [3]. Each sample is associated with a label from 0 to 9. Since this dataset is fully annotated, the ground truth semantic neighbors can be easily obtained based on the image labels. The entire dataset is partitioned into two parts: a training set with 69000 samples and a test set with 1000 samples. The training set is used for learning hashing functions and constructing the hashed look-up tables. For RBMs and SSH, we additionally randomly sample 2000 points from the training set and assign semantic nearest neighbor information (i.e. construct the pairwise label matrix $\mathbf{S}$ for SSH) based on the image labels. The gray-scale intensity values of images are directly used as features resulting in a 784-dim vector space. For LSH, we randomly select projections from a Gaussian distribution with zero-mean and identity covariance to construct the hash tables. For RBMs, we applied the similar neural network structure and parameters as discussed in [16]. The search results are inspected based on whether the returned images and the query share the same semantic labels. Hamming ranking based evaluation is used for quantitative performance measurement. The Hamming distance from the query and each sample in training dataset is computed and sorted. Then the preci-

[3] http://yann.lecun.com/exdb/mnist/

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on July 23,2020 at 03:17:34 UTC from IEEE Xplore. Restrictions apply.

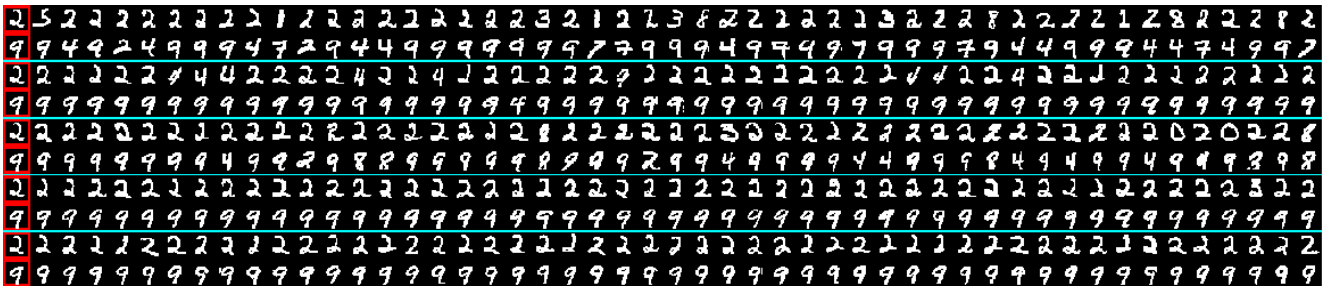


Figure 2. The 49 nearest neighbors of two example digits returned by different methods on the MNIST dataset. The left most digit with red bounding box is the query sample. From top to bottom (separated by color lines), the nearest neighbor samples are retrieved by LSH, SH, RBMs, and two variants of our method (SSH, and $\text{SSH}_{nonorth}$) using 48-bit codes.

sion on the top 500 samples is recorded. We vary the number of bits from 12 to 48 and show the performance curve in Figure 1 (a). From this figure, orthogonal SSH has superior performance for small number of bits (i.e. 8, 12, and 16 bits) because there is enough variance in top few orthogonal directions computed in SSH. But when using large number of bits, $\text{SSH}_{nonorth}$ performs the best. In addition, the precision-recall curve for 48-bit codes is shown in Figure 1 (b), which clearly shows that non-orthogonality helps $\text{SSH}_{nonorth}$ achieve the best performance over the entire Hamming ranking space when using a large number of bits. Figure 1 (c) provides the training time for different techniques. RBMs are the most expensive to train, needing roughly three orders of magnitude more time than the other methods. LSH needs negligible training time since the projections are randomly generated, instead of being learned. In terms of the query time, RBMs also need about 10 times more time to compute the binary codes through the trained neural network. SH method requires a little more time than the other three methods due to the calculation of the sinusoidal function. The code generation time can be ranked as: RBMs $\gg$ SH $>$ LSH $\simeq$ SSH $=$ $\text{SSH}_{nonorth}$. To visualize the quality of nearest neighbors, we show top neighbors retrieved by different techniques on two example digits in Figure 2. $\text{SSH}_{nonorth}$ tends to return more semantically consistent neighbors when using $48 - bit$ hashing codes.

4.2. One Million Gist Data

Next we apply our method on a large-scale application using a gist feature set sampled from the tiny images dataset [15], which has been used as a benchmark dataset for designing binary encoding approaches [8, 17, 16]. However, only a small portion of the dataset is manually labeled and the associated meta information is very noisy. We designed our experimental protocol for quantitative evaluation as below. A subset of one million data points is sampled to construct the training set and a separate set of 2000 samples is used as the test set. For SSH and RBMs, we randomly select 10000 samples from training set and compute a pairwise distance matrix $\mathbf{D}$ using L_2 norm. From $\mathbf{D}$, we derive the pairwise label matrix $\mathbf{S}$ using the following rule. The neighbor-pair is the pair of samples whose distance is within the $5th$ percentile of the whole set of distances in $\mathbf{D}$, and the nonneighbor-pair is the pair of samples whose distance is more than the $95th$ percentile. The $5th$ percentile distance threshold from a query is also used as the measurement of good neighbors.

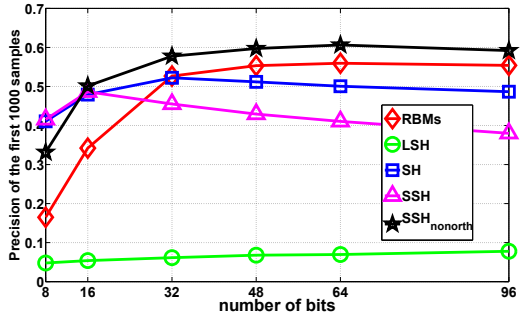
For quantitative evaluation, we compute the precision under two different scenarios, i.e. Hamming ranking and Hash lookup. For Hamming ranking based evaluation, the precision of the first 1000 retrieved samples is recorded. For Hash lookup evaluation, the ratios of good neighbors among the retrieved samples within hamming radius 2 are reported as the precision value. Moreover, if the query point does not have any returned samples within the Hamming ball of radius 2, it is treated as a failed query with precision zero. Figure 3 shows the results for different methods. SSH and $\text{SSH}_{nonorth}$ outperform the other methods in most of the cases with $\text{SSH}_{nonorth}$ performing significantly better than all the compared methods for higher bits as expected. Besides the quantitative evaluation, we also inspect some exemplar queries and retrieved nearest neighbors when using 48-bit hash codes (Figure 4). $\text{SSH}_{nonorth}$ tends to give better retrieval results with more visual relevance.

For the training cost, RBMs again have much longer training time (days) than that for all the other methods (around one minute). Particularly, the increase in training time from MNIST (70K) to Gist set (1 M) is nominal for the proposed SSH method, which shows that SSH is highly scalable.

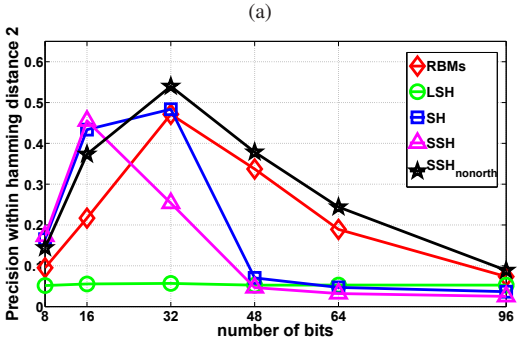
5. Conclusions and Future Work

In this paper, we have proposed a semi-supervised paradigm to learn efficient hash codes which can handle semantic similarity/dissimilarity among the data points. The proposed method combines empirical loss over the labeled data with other desirable properties e.g., balancing over both labeled and unlabeled data. The proposed method leads to a very simple eign-decomposition based solution which is extremely efficient. In fact, one can make the

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on July 23,2020 at 03:17:34 UTC from IEEE Xplore. Restrictions apply.



(a)



(b)

Figure 3. Experimental results on one million gist dataset. a): Precision of the first 1000 retrieved samples; b): Precision within hamming radius 2.

computations even faster by using iterative solvers to get top eigenvalues and eigenvectors. We further show that by relaxing the commonly used orthogonality constraints, one can achieve even better results, especially for larger number of bits. The experiments on two large datasets show superior performance of the proposed method over existing state-of-the-art techniques. In the future, we would like to explore the theoretical properties of the proposed SSH method. In particular, it will be useful to investigate if SSH can provide theoretical guarantees on the performance.

Acknowledgments

The authors would like to thank Dr. Zhenguo Li for his valuable comments. J. Wang was supported in part by Google Intern Scholarship. S.-F. Chang is supported in part by National Science Foundation Award CNS-07-51078.

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of ACM*, 45(6):891–923, 1998.

[2] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual Symposium on Computational Geometry*, pages 253–262. ACM, 2004.

[3] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

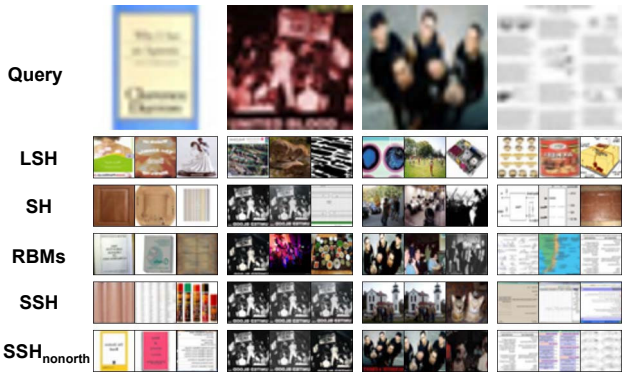


Figure 4. Top three nearest neighbors of four queries returned by different methods on one million Gist dataset. The images in the top row are the query samples. From top to bottom, the nearest neighbor samples retrieved by LSH, SH, RBMs, SSH, and SSH$_{nonorth}$) using 48-bit codes.

[4] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. of 25th VLDB*, 1999.

[5] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Proc. of NIPS*, volume 17, pages 513–520. 2005.

[6] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504, 2006.

[7] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *Proc. of the CVPR*, pages 1–8, 2008.

[8] B. Kulis and T. Darrell. Learning to Hash with Binary Reconstructive Embeddings. In *Proc. of NIPS*, volume 20, pages 1042–1050. 2009.

[9] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. *Proc. of ICCV*, 2009.

[10] N. Kumar, L. Zhang, and S. Nayar. What is a good nearest neighbors algorithm for finding similar patches in images? In *ECCV*, pages 364–378, 2008.

[11] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, 2009.

[12] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006.

[13] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc. of the ICCV*, 2003.

[14] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*, pages 1–8, 2008.

[15] A. Torralba, R. Fergus, W. Freeman, and C. MIT. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. on PAMI*, 30(11):1958–1970, 2008.

[16] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proc. of CVPR*, pages 1–8, 2008.

[17] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. of NIPS*, volume 21, pages 1753–1760. 2008.

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on July 23,2020 at 03:17:34 UTC from IEEE Xplore. Restrictions apply.