Topology Preserving Hashing for Similarity Search

Lei Zhang[1,2], Yongdong Zhang[2], Jinhui Tang[3], Xiaoguang Gu[2], Jintao Li[2], Qi Tian[4]
[1]University of Chinese Academy of Sciences
[2]Institute of Computing Technology,Chinese Academy Of Sciences
[3]Nanjing University of Science and Technology
[4]University of Texas at San Antonio
{zhanglei09,zhyd,xggu,jtli}@ict.ac.cn, jinhuitang@mail.njust.edu.cn, qitian@cs.utsa.edu

ABSTRACT

Binary hashing has been widely used for efficient similarity search. Learning efficient codes has become a research focus and it is still a challenge. In many cases, the real-world data often lies on a low-dimensional manifold, which should be taken into account to capture meaningful neighbors with hashing. The importance of a manifold is its topology, which represents the neighborhood relationships between its subregions and the relative proximities between the neighbors of each subregion, e.g. the relative ranking of neighbors of each subregion. Most existing hashing methods try to preserve the neighborhood relationships by mapping similar points to close codes, while ignoring the neighborhood rankings. Moreover, most hashing methods lack in providing a good ranking for query results since they use Hamming distance as the similarity metric, and in practice, there are often a lot of results sharing the same distance to a query. In this paper, we propose a novel hashing method to solve these two issues jointly. The proposed method is referred to as *Topology Preserving Hashing* (TPH). TPH is distinct from prior works by preserving the neighborhood rankings of data points in Hamming space. The learning stage of TPH is formulated as a generalized eigendecomposition problem with closed form solutions. Experimental comparisons with other state-of-the-art methods on three noted image benchmarks demonstrate the efficacy of the proposed method.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Content Analysis and Indexing, Information Search and Retrieval

General Terms

Algorithms, Performance

Keywords

Similarity Search, Approximate Nearest Neighbor Search, Binary Hashing, Topology Preserving Hashing.

MM'13, October 21–25, 2013, Barcelona, Spain.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2404-5/13/10 ...$15.00.
http://dx.doi.org/10.1145/2502081.2502091.

1. INTRODUCTION

High-dimensional similarity search is a fundamental problem in many content-based search systems and also widely exists in many related application areas, such as machine learning, computer vision, information retrieval and data mining. Due to the linear time complexity, traditional linear nearest neighbor (NN) search is computationally prohibitive, especially when dataset contains millions or even billions of data points. To solve this problem efficiently, a number of methods have been proposed, such as KD-Tree [2] and Locality Sensitive Hashing (LSH) [1]. Recently, binary hashing [22, 29, 14, 4, 23, 13, 12, 10, 3] is becoming increasingly popular for efficient approximate nearest neighbor (ANN) search due to its good query and storage efficiency.

Given a dataset, binary hashing methods map each data point $\boldsymbol{x}$ to a binary code $H(\boldsymbol{x})$ and perform bit-wise operations to search neighbors. To capture meaningful neighbors, the neighborhood structure should be preserved after Hamming embedding. In many cases, the real-world dataset often lies on a low-dimensional manifold embedded in a high-dimensional space [14]. To characterize a manifold, its topology, which represents the neighborhood relationships between subregions and the relative proximities between the neighbors of subregions, is essential [11]. A lot of hashing methods [8, 29, 3, 14, 13, 12] have been developed to preserve the neighborhood relationships (we call neighborhood-preserving) by mapping similar points to close binary codes. In these methods, the neighborhood relationships are taken into account in hash function learning, however, the relative proximities between the neighbors, e.g. the rankings, are not. As a result, these methods cannot preserve the data topology well. More precisely, as shown in Fig. 1(b), for a data point $\boldsymbol{p}$ and its neighbors $\boldsymbol{q}_1$, $\boldsymbol{q}_2$, $H(\boldsymbol{q}_1)$ and $H(\boldsymbol{q}_2)$ are still neighbors of $H(\boldsymbol{p})$, however, the ranking between $H(\boldsymbol{q}_1)$ and $H(\boldsymbol{q}_2)$ is not preserved.

To preserve the local topology of a dataset, a straightforward way is to preserve the original distance after Hamming embedding. Many hashing methods [1, 9, 20, 10] have been developed (we call distance-preserving) and in these methods, the Hamming distance between binary codes are used as a reconstruction of the distance between data points in the original data space or a kernel space. Since the Hamming distance is bounded by the code length, the original distance to Hamming distance mapping is many-to-one, which indicates the reconstruction is far from optimal. On the other hand, characterizing a manifold with distances turns out to support and bolt it with rigid steel beam [11]. In many cases, the optimal embedding of a manifold needs some flexibili-

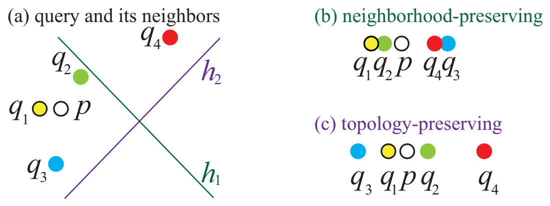


Figure 1: The essence of our proposed TPH. (a) A data point p and its neighbors. (b) In neighborhood-preserving hashing (h_1), the neighborhood relationship is preserved, but the ranking of neighbors is not. (c) In topology-preserving hashing (h_2), besides the neighborhood relationship, the ranking is also preserved. The hash functions are depicted as linear projections for the sake of illustration.

ty: some subregions should be locally stretched or shrunk to embed them in a low-dimensional space within which the topology can be well preserved. Moreover, most distance functions make no distinction between the manifold and its surrounding empty space, to some extent, the exact measure of distance depends not only on the manifold itself, but also, for a large part, on a given embedding of the manifold. As a result, although designed for distance-preserving, these methods [9, 10] still cannot preserve the local topology well.

Another limitation of most hashing methods is that these methods simply adopt the Hamming distance as distance metric. Since Hamming distance is bounded by the code length, there can be a lot of results sharing the same distance to a query in practice, posing a critical issue for similarity search, e.g. k nearest neighbor search, where ranking is important. Weighted Hamming distance [7, 30] have been developed to alleviate this limitation, however, they are all post-processing algorithms while the ranking information is already lost too much after embedding. Apparently, if the ranking information can be encoded in binary codes, this limitation would be better alleviated.

In this paper, we propose a Topology Preserving Hashing (TPH) method to solve the above two issues jointly. As stated in [11], a manifold can be entirely characterized by giving the relative proximities between its subregions. Actually, comparative information between distance, like inequalities or rankings, suffices to characterize a manifold for any embeddings. In this work, the relative proximities are revealed by the distances, i.e. the ranking of neighbors of each subregion. Therefore, the essence of our Topology Preserving Hashing is to not only preserve the neighborhood relationships, but also preserve the neighborhood rankings. As shown in Fig. 1(c), TPH ensures that $H(\boldsymbol{q}_1)$ and $H(\boldsymbol{q}_2)$ are neighbors of $H(\boldsymbol{p})$, and $H(\boldsymbol{q}_1)$ is still a nearer neighbor of $H(\boldsymbol{p})$ in Hamming space. Our method can be considered as a tight version of neighborhood-preserving and a loose version of distance-preserving. The main contributions of this paper are briefly outlined as follows:

(1) We propose a novel hashing method to learn the Hamming embeddings for a dataset, such that not only the neighborhood relationships between data points, but also the neighborhood ranking of each data point, is preserved after embedding. To the best of our knowledge, this is one of the first work that tries to incorporate neighborhood ranking information with hashing.

(2) Experimental results demonstrate the superiority of our method as compared with other state-of-the-art methods. Moreover, the limitation of most hashing methods, which is lacking in providing a good ranking for query results, is also well alleviated.

(3) The label information of a dataset can be easily leveraged in the learning stage of our method, extending it to a semi-supervised method to capture semantic neighbors in a simple manner.

The rest of this paper is organized as follows. The related work is discussed in Section 2. The Topology Preserving Hashing method is proposed in Section 3. Section 4 describes our experiments and Section 5 concludes this paper.

2. RELATED WORK

With the proliferation of various kinds of data, e.g. music, image and video, in content-based search systems, fast similarity search has attracted a significant attention. One classical kind of methods to address this problem is the tree-based index, e.g. KD-Tree [2]. However, this kind of methods cannot work well for high-dimensional data. Their performances degrade significantly to linear scan as the dimensionality increases. Moreover, most of visual descriptors, like SIFT [15] and GIST [18], are high-dimensional vectors. Therefore, tree-based indexes are not preferable in high-dimensional similarity search problems. Another kind of ANN search algorithm is based on vector quantization, such as kmeans LSH [19] and Product Quantization (PQ) [5]. The key of these methods is the compositionality. In PQ, by dividing each data into several subspaces and expressing data in terms of recurring parts, the representational capacity of PQ grows exponentially in the number of subspaces. In these methods [19, 5], each data point is represented by a reconstructed cluster center and the search process is performed in the original data space. As a result, the search process is time-consuming even with a inverted file indexing [5]. Recently, hashing based methods have been widely used for efficient similarity search [31, 23, 32] since it allows constant-time search. A lot of hashing methods have been proposed, and these methods can be roughly divided into two main categories [3, 14]: data-independent methods and data-dependent methods.

One representative kind of data-independent methods is hashing with random projections, such as Hamming Embedding [4] and Locality Sensitive Hashing (LSH) [1, 8, 6]. In these methods, the hash functions are random projections which are independent of dataset. Theoretically, it is guaranteed that the original distance or similarity are asymptotically preserved in Hamming space with increasing code length, hence LSH-related methods usually require long codes to achieve good precision. However, long codes result in low recall since the collision probability of similar points mapped to close binary codes decreases exponentially as the code length increases. As a result, LSH-related methods usually construct multi-tables to ensure a reasonable probability that a query will collide with its near neighbors in at least one table, which leads to a long query time and a high memory occupation. Another representative method is Shift Invariant Kernel Hashing (SIKH) [20]. It is a distribution-free method based on random features mapping for shift-invariant kernels, and the expected Hamming distance between the binary codes is related to the distance between data points in a kernel space. Similar to LSH, SIKH also needs relatively long codes to ensure good performance [3]. As a result, in practice, data-independent methods are often less effective than data-dependent methods.

To generate more compact codes, many data-dependent methods have been developed to learn hash functions from dataset. Semantic Hashing [22] adopts a deep generative model based on restricted Boltzmann machine to learn hash functions. In PCA-Hashing [28], the eigenvectors corresponding to the largest eigenvalues of the data covariance matrix are used to form a projection matrix for hashing. The spectral graph partitioning strategy is employed to develop new kinds of hashing methods, e.g. Spectral Hashing (SPH) [29, 12]. SPH uses the simple analytical eigenfunction solution of 1-D Laplacians as the hash function. In SPH [29], two important constraints on binary codes learning are introduced: (1) each hash bit balanced partitions the dataset; (2) different hash bits should be uncorrelated. These two constraints are widely accepted in many research works [26, 14, 3]. In [3], Iterative Quantization (ITQ) is proposed to learn an orthogonal rotation matrix to refine the initial P-CA projection matrix [28] to minimize the quantization error of mapping the data from original data space to Hamming space. To minimize the reconstruction error between the o-riginal distance of data points and the Hamming distances of the corresponding codes, Binary Reconstruction Embedding (BRE) is proposed in [9]. In BRE, the Hamming distance is used as an reconstruction of the distance between data points in the original data space. In [10], the original LSH is generalized to a kernel space, called Kernelized Locality Sensitive Hashing (KLSH). KLSH is proposed to address the limitation that the original LSH methods cannot apply for high-dimensional kernelized data when the underlying feature embedding for the kernel is unknown. In KLSH, the Hamming distance is an approximation of the distance between data points in a kernel space. To exploit the spectral properties of the data affinity to generate better binary codes, many other algorithms, such as Semi-Supervised Hashing (SSH) [25, 26], Anchor Graph Hashing (AGH) [14], LDAHash [24] and Kernel-Based Supervised Hashing (KSH) [13], have been developed. Both labeled and unlabeled data are used in SSH [26] to learn hash functions to minimize the empirical error on the labeled data and maximize the information entropy regularization on all data. AGH [14] applies the similar formulation of SPH [29] for hash codes generation, while its neighborhood graph is constructed in a novel way such that it can be applied to large-scale dataset. Provided a set of positive and negative data pairs, LDA-Hash [24] learns the hash functions in a supervised manner by making the Hamming distance minimized between positive pairs and maximized between negative pairs. In [13], by leveraging label information and using the equivalence between optimizing the code inner products and the Hamming distances, KSH could map a dataset to compact codes whose Hamming distances are minimized on similar pairs and simultaneously maximized on dissimilar pairs.

In most existing binary hashing methods, including those methods discussed above, to capture meaningful neighbors, the neighborhood structure (i.e. local topology) of a dataset should be preserved after Hamming embedding. Mostly, the real-world data often lies on a low-dimensional manifold and to characterize a manifold, the neighborhood relationships between data points and the relative proximities between the neighbors of data points are both essential [11]. However, in most hashing methods [28, 29, 14, 3, 24], the neighborhood relationships are preserved, while the relative proximities are not guaranteed to be preserved. Even in the hashing methods [1, 20, 9, 10] developed for distance preserving, the topology is still not well preserved due to the fact that the original distance reconstruction using Hamming distance is far from optimal. Therefore, the learned codes of these methods are not the most optimal embedding for a dataset. In this paper, we propose a novel hashing method to solve the above problems. Besides the neighborhood relationships, the relative proximities between the neighbors of data points are also taken into account in the hash function learning process.

3. TOPOLOGY PRESERVING HASHING

This section describes our Topology Preserving Hashing (TPH) method. First, we introduce the motivation of topology preserving. Then, we give the deduction of TPH and formulate its learning stage as an optimization problem with closed form solutions. After that, we discuss some aspects of TPH which can be further developed and leave them for our future work. Some notations are given below to facilitate our discussion.

Given a dataset consists of n data points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \in \mathbb{R}^d$ that form the columns of the data matrix $X \in \mathbb{R}^{d\times n}$, the paradigm of binary hashing is to first use a set of linear or non-linear hash functions $F = \{f_k : \mathbb{R}^d \to \mathbb{R}\}_{k=1}^m$ to map each $\boldsymbol{x}_i$ to $F(\boldsymbol{x}_i) \in \mathbb{R}^m$, and then binarize $F(\boldsymbol{x}_i)$ to embed $\boldsymbol{x}_i$ into the m-dimensional Hamming space[1] $\mathbb{H}^m \in \{-1, 1\}^m$. The binary code of $\boldsymbol{x}_i$ is denoted as $\boldsymbol{y}_i = H(\boldsymbol{x}_i)$ and forms the i-th column of the code matrix $Y \in \{-1, 1\}^{m\times n}$. In this paper, we restrict our attention to affine embedding:

$$H(\boldsymbol{x}) = \operatorname{sgn}(W^T\boldsymbol{x} + \boldsymbol{t}) \tag{1}$$

where W is a $d \times m$ projection matrix with each column the projection vector $\boldsymbol{\omega}_k$, and $\boldsymbol{t}$ is a $m \times 1$ vector. Without loss of generality, we could assume the data are zero-centered, i.e. $\sum_{i=1}^n \boldsymbol{x}_i = \boldsymbol{0}$, thus $\boldsymbol{t}$ is simply set to $\boldsymbol{0}$ and each hash function is given by $h_k(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{\omega}_k^T\boldsymbol{x})$. Therefore, we can write the entire encoding process as $Y = \operatorname{sgn}(W^TX)$, where $\operatorname{sgn}(W^TX)$ is the matrix of signs of individual elements.

3.1 Motivation

As the real-world data often lies on a low-dimensional manifold [14], to capture meaningful neighbors with hashing, the neighborhood structure of the manifold should be preserved. To characterize a manifold, its topology, which represents the neighborhood relationships between subregions and the relative proximities between neighbors of each subregion, is essential [11]. As a manifold can be entirely characterized by giving the relative or comparative proximities: a first region is close to a second one but far from a third one, comparative information between distances, e.g. rankings, suffices to characterize a manifold. Therefore, more effective Hamming embeddings can be learned by taking account of the data topology in the learning stage. In this work, the relative proximities between neighbors of a data point are revealed by distances, i.e. the ranking of neighbors. Apparently, if the local topology is well preserved in Hamming

[1]We treat '0' bit as '-1' in our deduction and use '0' back in data embedding and query hashing. The conversion form -1/1 code to 0/1 code is trivial.

space, the ambiguity caused by ranking with Hamming distance can be well alleviated. As a result, topology preserving hashing is promising and our goal is to learn binary codes such that the local topology of a dataset is well preserved after the dataset has been embedded into Hamming space.

3.2 Topology Preserving Projection Learning

We begin with the definition of *local topology preserving* [16]: For data points $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s, \boldsymbol{x}_t$ in a metric space $(\mathcal{M}, d_{\mathcal{M}})$, where $d_{\mathcal{M}}(\cdot,\cdot)$ parametrizes the dissimilarity or distance in $\mathcal{M}$, and their Hamming embeddings $\boldsymbol{y}_i, \boldsymbol{y}_j, \boldsymbol{y}_s, \boldsymbol{y}_t$ generated by hash function $H(\boldsymbol{x})$. $H(\boldsymbol{x})$ is called topology preserving if the following condition holds:

$$\text{if } d_{\mathcal{M}}(\boldsymbol{x}_i,\boldsymbol{x}_j) \le d_{\mathcal{M}}(\boldsymbol{x}_s,\boldsymbol{x}_t), \text{ then } d_H(\boldsymbol{y}_i,\boldsymbol{y}_j) \le d_H(\boldsymbol{y}_s,\boldsymbol{y}_t) \tag{2}$$

where $d_H(\boldsymbol{y}_1,\boldsymbol{y}_2) = \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2$ is the Hamming distance[2] between binary codes $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. This definition says that for a data point $\boldsymbol{p}$ and its neighbors $\boldsymbol{q}_1, \boldsymbol{q}_2$, if $\boldsymbol{q}_1$ is a nearer neighbor of $\boldsymbol{p}$, then $H(\boldsymbol{q}_1)$ is also a nearer neighbor of $H(\boldsymbol{p})$, which means the local topology of the neighborhood of a data point in the original data space is well preserved in Hamming space by hashing with $H(\boldsymbol{x})$.

Given $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \in \mathbb{R}^d$ in a metric space $(\mathcal{M}, d_{\mathcal{M}})$, our goal is to learn a set of m-bit Hamming embeddings $\{\boldsymbol{y}_i\}_{i=1}^n \in \{-1,1\}^m$ in a topology-preserving manner. We define the following objective function measuring the empirical accuracy of topology preserving:

$$\mathcal{O}(Y) = \frac{1}{2}\sum_{i,j,s,t} \operatorname{sgn}\left(d_{\mathcal{M}}(\boldsymbol{x}_i,\boldsymbol{x}_j) - d_{\mathcal{M}}(\boldsymbol{x}_s,\boldsymbol{x}_t)\right) \operatorname{sgn}\left(d_H(\boldsymbol{y}_i,\boldsymbol{y}_j) - d_H(\boldsymbol{y}_s,\boldsymbol{y}_t)\right) \tag{3}$$

Clearly, for two pairs $(\boldsymbol{x}_i,\boldsymbol{x}_j)$ and $(\boldsymbol{x}_s,\boldsymbol{x}_t)$, the summed term in Eq. (3) would be 1 if the relative proximity between $(\boldsymbol{y}_i,\boldsymbol{y}_j)$ and $(\boldsymbol{y}_s,\boldsymbol{y}_t)$ is consistent with that between $(\boldsymbol{x}_i,\boldsymbol{x}_j)$ and $(\boldsymbol{x}_s,\boldsymbol{x}_t)$, or -1 otherwise. Hence, $\mathcal{O}(Y)$ is a reasonable measure function. We replace the signs of summed terms with their signed magnitudes. This relaxation is quite intuitive in the sense that for a data point $\boldsymbol{p}$ and its neighbors $\boldsymbol{q}_1, \boldsymbol{q}_2$, it not only desires $d_{\mathcal{M}}(\boldsymbol{p},\boldsymbol{q}_1) - d_{\mathcal{M}}(\boldsymbol{p},\boldsymbol{q}_2)$ and $d_H(\boldsymbol{p},\boldsymbol{q}_1) - d_H(\boldsymbol{p},\boldsymbol{q}_2)$ to have the same signs in order to preserve the local topology, but also the larger $d_{\mathcal{M}}(\boldsymbol{p},\boldsymbol{q}_1)$ than $d_{\mathcal{M}}(\boldsymbol{p},\boldsymbol{q}_2)$, the larger $d_H(\boldsymbol{p},\boldsymbol{q}_1)$ than $d_H(\boldsymbol{p},\boldsymbol{q}_2)$. Meanwhile, if $d_{\mathcal{M}}(\boldsymbol{p},\boldsymbol{q}_1) - d_{\mathcal{M}}(\boldsymbol{p},\boldsymbol{q}_2)$ and $d_H(\boldsymbol{p},\boldsymbol{q}_1) - d_H(\boldsymbol{p},\boldsymbol{q}_2)$ have different signs, it also desires $|d_H(\boldsymbol{p},\boldsymbol{q}_1) - d_H(\boldsymbol{p},\boldsymbol{q}_2)|$ small in order to alleviate the local topology deviation. For brevity, we denote $d_{\mathcal{M}}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ as d_{ij} and $d_H(\boldsymbol{y}_i,\boldsymbol{y}_j)$ as h_{ij} in the following. With this relaxation, $\mathcal{O}(Y)$ (3) becomes:

$$\mathcal{O}(Y) = \frac{1}{2}\sum (d_{ij} - d_{st})(h_{ij} - h_{st}) \tag{4}$$

Apparently, since d_{ij} is predefined, the most optimal solution Y is that makes the ranking of h_{ij} in $\{h_{ij}\}_{i,j=1}^n$ identical with that of d_{ij} in $\{d_{ij}\}_{i,j=1}^n$. As a result, based on the Rearrangement Inequality, the optimal Y will also maximize the following function:

$$\mathcal{T}(Y) = \frac{1}{2}\sum_{ij} d_{ij}h_{ij} = \frac{1}{2}\sum_{i,j} d_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 \tag{5}$$

[2]The actual Hamming distance is $\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2/4$ when $\boldsymbol{y} \in \{-1,1\}^m$. For brevity, we simply use $\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2$ to measure the Hamming distance.

Moreover, since $\mathcal{O}(Y)$ can be rewritten into the difference of two parts:

$$\mathcal{O}(Y) = \frac{1}{2}\sum (d_{ij}h_{ij} + d_{st}h_{st}) - \frac{1}{2}\sum (d_{ij}h_{st} + d_{st}h_{ij}) \tag{6}$$

Maximize $\mathcal{T}(Y)$ leads to maximizing the first part of $\mathcal{O}(Y)$, and based on the Rearrangement Inequality, the second part of $\mathcal{O}(Y)$ will not be maximized when $\mathcal{T}(Y)$ is maximized. Moreover, in view of the fact that the number of summed terms in $\mathcal{T}(Y)$ is $O(n^2)$, much less than that of $\mathcal{O}(Y)$, $O(n^4)$, instead of maximizing $\mathcal{O}(Y)$ directly, we try to maximize $\mathcal{T}(Y)$ alternatively. As a result, the optimization problem for solving Y is formulated as:

$$Y = \arg\max_{Y\in\{-1,1\}^{m\times n}} \mathcal{T}(Y) \tag{7}$$

In $\mathcal{T}(Y)$, d_{ij} weights each summed term $\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$, thus we call it the **topo-weight** and denote it as τ_{ij}, as not only the original distance d_{ij} can be used as a topo-weight. More details about topo-weight is given in Section 3.3. By defining a **Topo-Weight Matrix** $S_t \in \mathbb{R}^{n\times n}$, $S_t(i,j) = \tau_{ij}$, problem (7) can be rewritten as:

$$Y = \arg\max_{Y\in\{-1,1\}^{m\times n}} \operatorname{Tr}\{Y\varGamma_t Y^T\} \tag{8}$$

where $\varGamma_t = D_t - S_t$, $D_t = \operatorname{diag}(S_t\mathbf{1}_{n\times 1})$. In the following, we call $\varGamma_t$ the **Topo-Training Matrix**.

However, maximizing $\mathcal{T}(Y) = \operatorname{Tr}\{Y\varGamma_t Y^T\}$ is not easy to achieve because it is neither convex nor smooth. Motivated by the spectral methods for hashing [29, 14], we adopt the **Spectral Relaxation** [13] scheme to approximately maximize $\mathcal{T}(Y)$. After substituting Y with $\operatorname{sgn}(W^TX)$ and applying the spectral relaxation trick to drop the sign functions involved in $\mathcal{T}(Y)$, (8) becomes a quadratic optimization problem for solving W:

$$W = \arg\max_{W\in\mathbb{R}^{d\times m}} \mathcal{T}(W) \tag{9}$$

where $\mathcal{T}(W) = \operatorname{Tr}\{W^TX\varGamma_tX^TW\}$.

Maximizing $\mathcal{T}(W)$ is to preserve the neighborhood ranking of each data point. To preserve the data topology, we also have to preserve the neighborhood relationships as done in [29, 14, 23]. Suppose w_{ij} is the similarity between $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ in the original data space. We call w_{ij} a **simi-weight** and define a **Simi-Weight Matrix** $S_s \in \mathbb{R}^{n\times n}$, $S_s(i,j) = w_{ij}$. The problem of solving W for neighborhood preserving is formulated as [29, 14]:

$$W = \arg\min_{W\in\mathbb{R}^{d\times m}} \frac{1}{2}\sum_{ij} w_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 \tag{10}$$

Applying the same spectral relaxation trick to problem (10), we have another quadratic optimization problem:

$$W = \arg\min_{W\in\mathbb{R}^{d\times m}} \mathcal{D}(W) \tag{11}$$

where $\mathcal{D}(W) = \operatorname{Tr}\{W^TX\varGamma_sX^TW\}$, $\varGamma_s = D_s - S_s$, $D_s = \operatorname{diag}(S_s\mathbf{1}_{n\times 1})$. We call $\varGamma_s$ the **Simi-Training Matrix**.

In summary, the optimal projection matrix W should not only maximize $\mathcal{T}(W)$ (9), but also minimize $\mathcal{D}(W)$ (11). Hence, given a dataset X, we have two approaches to learn the optimal projection matrix W. It can be learned by maximizing the following two objective functions:

$$\mathcal{O}_1(W) = \mathcal{T}(W) - \beta\mathcal{D}(W) \tag{12}$$

$$\mathcal{O}_2(W) = \frac{\mathcal{T}(W)}{\mathcal{D}(W)} \tag{13}$$

In $\mathcal{O}_1(W)$, β is a positive scalar, which relatively weights the neighborhood-preserving term $\mathcal{D}(W)$. In this work, $\mathcal{O}_2(W)$ is used as the objective function as it somewhat resembles the spirit of linear discriminant analysis (LDA) [24] and has no regularization weight.

However, the current learning stage cannot scale well with the size of training set, as the Topo-Training Matrix $\varGamma_t$ and Simi-Training Matrix $\varGamma_s$ are both $n \times n$. Therefore, in practice, only a subset of X, X_t, is used as the training set to construct $\varGamma_t$ and $\varGamma_s$. As a result, the objective function $\mathcal{O}_2(W)$ (13) measures only the empirical accuracy on X_t and is prone to overfitting especially when the size of X_t is small compared to the entire dataset X. To get better generalization ability, we add a regularization by incorporating conditions that lead to desirable properties of hash codes, independent of the performance on the training set. Here, the variances of hash bits are used as the regularization [26]. The overall learning stage of topology preserving hashing is formulated as the following optimization problem:

$$W = \arg\max_{W\in\mathbb{R}^{d\times m}} \mathcal{O}(W) \tag{14}$$

where

$$\begin{aligned} \mathcal{O}(W) &= \frac{\mathrm{Tr}\{W^T A W\}}{\mathrm{Tr}\{W^T B W\}} \\ A &= X_t \varGamma_t X_t^T + \alpha X X^T \\ B &= X_t \varGamma_s X_t^T \end{aligned} \tag{15}$$

α is a positive scalar relatively weights the regularization $\mathrm{Tr}\{W^T X X^T W\}$. In $\mathcal{O}(W)$, a column $\boldsymbol{\omega}_k$ of W with large norm makes its associated $\boldsymbol{\omega}_k^T A \boldsymbol{\omega}_k$ and $\boldsymbol{\omega}_k^T B \boldsymbol{\omega}_k$ the dominant factors in the numerator and denominator of $\mathcal{O}(W)$, thus affecting the actual maximum of $\mathcal{O}(W)$. Furthermore, in our hashing method, the direction of $\boldsymbol{\omega}_k$ is more important than its norm. As a result, we impose a "generalized normalization constraint" on W, which is $\boldsymbol{\omega}_k^T B \boldsymbol{\omega}_k = 1$. By adding this constraint, problem (14) is equivalent to:

$$\begin{aligned} W &= \arg\max_{W=[\boldsymbol{\omega}_1,\boldsymbol{\omega}_2,\cdots,\boldsymbol{\omega}_m]} \mathrm{Tr}(W^T A W) \\ \text{s.t.} &\quad \boldsymbol{\omega}_k^T B \boldsymbol{\omega}_k = 1,\ \forall k \in [1, m] \end{aligned} \tag{16}$$

Note that, in most hashing methods [29, 26, 14], two important constraints are often imposed on hash function learning. One is all hash bits are independent (this constraint is often relaxed to hash bit decorrelation, i.e. $YY^T = nI_m$), and the other one is each bit maximizes its information entropy by balanced partitioning the dataset, i.e. $Y\mathbf{1}_{n\times 1} = \mathbf{0}$. However, these two constraints are not imposed on (16). In the following, we will show that, by using the solutions of (16) as hash functions, the learned binary codes Y somewhat satisfy these two constraints implicitly.

Problem (16) can be easily solved and the optimal solution W is given by the m generalized eigenvectors corresponding to the top m largest generalized eigenvalues of the following generalized eigendecomposition problem:

$$AW = BW\Lambda \tag{17}$$

In most cases, A and B are both positive definite Hermitian matrices. Therefore, eigenvectors $\boldsymbol{\omega}_k$ corresponding to different eigenvalues are linearly independent. Moreover, we also have $\boldsymbol{\omega}_i^T B \boldsymbol{\omega}_j = 0, \forall i \neq j$, which means W is weighted orthogonal with B, i.e. $W^T B W = I_m$. This can be viewed as a generalized orthogonality constraint on W. Since the orthogonality on projection matrix W is a relaxation of hash bit decorrelation [26], the learned binary codes of our method somewhat satisfies the decorrelation constraint. Meanwhile, by adding the regularization $\mathrm{Tr}(W^T X X^T W)$ with all data points to $\mathcal{O}(W)$, the balanced partition constraint is also intrinsically satisfied when X is zero-centered as proved in [26]. As a result, the decorrelation and balanced partition constraints are not imposed on problem (16).

After learning m projection vectors, one straightforward hashing method is using $h_k(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{\omega}_k^T \boldsymbol{x})$ as the hash function to first project X on W and then binarize $Z = W^T X$ to embed X into Hamming space. This method is denoted as **TPH** in our experiments. However, the latter quantization step may distort the learned local topology, hence we have to find another way to binarize Z. It is easy to see that $\mathrm{sgn}(\boldsymbol{v})$ is the vertex of the hypercube $\{-1, 1\}^m$ closest to $\boldsymbol{v}$ in terms of Euclidean distance. To preserve the learned local topology well in Hamming space, a small quantization loss $\|\mathrm{sgn}(\boldsymbol{v}) - \boldsymbol{v}\|^2$ is desirable. One way to achieve this is to rotate $\boldsymbol{v}$ and map it to the nearest vertex [3]. Note that, for any orthogonal matrix $R \in \mathbb{R}^{m\times m}$, $\mathrm{Tr}\{RW^T A W R^T\} = \mathrm{Tr}\{W^T A W\}$ and $\mathrm{Tr}\{RW^T B W R^T\} = \mathrm{Tr}\{W^T B W\}$. Therefore, we are free to orthogonal transformation to rotate the projected data Z to minimize the following quantization loss [3]:

$$\mathcal{Q}(Y, R) = \|Y - RZ\|_F^2 = \|Y^T - Z^T R^T\|_F \tag{18}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Here, we use the iterative quantization procedure proposed in [3] to find a local minimum of $\mathcal{Q}(Y, R)$. It consists of two steps in each iteration: **fix** R and **update** Y; **fix** Y and **update** R. In the first step, Y is simply set to $\mathrm{sgn}(RZ)$. In the second step, minimization of $\mathcal{Q}(Y, R)$ is the classical Orthogonal Procrustes problem, the optimal R can be easily solved by computing the singular value decomposition of $ZY^T = U\Sigma V^T$ and setting $R^T = UV^T$, i.e. $R = VU^T$. For more details of this procedure, please refer to [3]. The final projection matrix is set to WR^T and the embedding of X is given by $\mathrm{sgn}(RW^T X)$. Hashing with the rotated projection matrix is denoted as **TPH-R** in our experiment, the whole procedure of TPH-R is outlined in Algorithm 1. Note that, TPH uses the projection matrix W learned in Step 2 of Algorithm 1 for hashing, thus its algorithm is not outlined.

3.3 Discussion

In the deduction of TPH, we call d_{ij} in $\mathcal{T}(Y)$ (5) topo-weight as it indicates the relative proximities between the neighbors of a data point, i.e. the data point is closer to one neighbor than another. Based on the Rearrangement Inequality, when maximizing $\mathcal{T}(Y)$, small d_{ij} yields small $\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$ while large d_{ij} yields large $\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$. Therefore, to preserve the local topology, the more similar a data pair, the smaller its topo-weight. Any $\tau_{ij} = f(d_{ij})$ can be used as the topo-weight when $f(\cdot)$ is monotonically non-decreasing w.r.t. d_{ij}. As a result, with metric learning method [27], TPH can be applied to practical applications where the similarity metrics beyond Euclidean distance used. In our ex-

```
Algorithm 1: Topology Preserving Hashing (TPH-R)
  Input: Dataset $X$ (zero-centered), Training set
         $X_t \subset X$, $\alpha$ the regularization weight and $m$ the
         number of hash bits.
  Output: Topology-preserving binary codes $Y$ and hash
          function $H(\cdot)$.
1 Construct the Topo-Training Matrix $\mathit{\Gamma}_t$ and
  Simi-Training Matrix $\mathit{\Gamma}_s$;
2 Solve the generalized eigenvalue decomposition
      $\left[X_t \mathit{\Gamma}_t X_t^T + \alpha X X^T\right] W = \left[X_t \mathit{\Gamma}_s X_t^T\right] W \Lambda$
  to determine $m$ eigenvectors corresponding to the
  top-$m$ largest generalized eigenvalue. Set the projection
  matrix $W = [\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_m] \in \mathbb{R}^{d \times m}$;
3 Use the iterative quantization procedure proposed in [3]
  to get an optimal rotation matrix $R$ such that the
  quantization loss $\mathcal{Q}(Y, R)$ (18) converges to a local
  minimum;
4 The rotated projection matrix is given by $P = WR^T$.
  The hash function of TPH-R is
      $H(\boldsymbol{x}) = \mathrm{sgn}(P^T \boldsymbol{x})$
  and the hamming embedding of $X$ is $Y = \mathrm{sgn}(P^T X)$.
```

periments, we introduce two weighting schemes based on the similarity metric of original data space.

Although our TPH in this work is unsupervised, it can be extended to a semi-supervised method by adopting the following label-based weighting scheme. In the construction of Topo-Training Matrix $\mathit{\Gamma}_t$, if the label of each data point (e.g. semantic/class label) or pairwise relationship (similar or dissimilar) of each data pair is provided, then for a neighbor pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, the associated topo-weight τ_{ij} is set to a small value, while for a non-neighbor pair, τ_{ij} is set to a relatively large value. By the same token, the label information can also be leveraged in the construction of Simi-Training Matrix $\mathit{\Gamma}_s$. By adopting this label-based weighting scheme, TPH is allowed to learn the projection matrix W in a semi-supervised manner. Experimental results in Section 4.4 demonstrate the effectiveness of adopting this label-based weighting scheme for semantic neighbor search.

In this paper, the simple linear projection is used as the hash function (1) and it still gives good similarity search performance in our experiments. Intuitively, different kinds of hash functions give rise to various hashing algorithms. In [9, 13], hash function with kernel κ plugged is given by

$$h_k(\boldsymbol{x}) = \mathrm{sgn}\left(\sum_{j=1}^{s} w_{kj}\kappa(\boldsymbol{a}_{kj}, \boldsymbol{x}) - b_k\right) \tag{19}$$

where $\boldsymbol{a}_{k1}, \cdots, \boldsymbol{a}_{ks}$ randomly selected from dataset are s kernel points of hash function $h_k(\cdot)$, $\boldsymbol{\omega}_k = (w_{k1}, \cdots, w_{ks})^T$ is the coefficient vector and $b_k = \sum_{i=1}^{n}\sum_{j=1}^{m}\kappa(\boldsymbol{a}_{kj}, \boldsymbol{x}_i)/n$ is the bias. If the set of kernel points of each hash function is identical, this embedding can be written more compactly in a matrix form:

$$Y = \mathrm{sgn}\left(W^T\left[K(\boldsymbol{x}_1), \cdots, K(\boldsymbol{x}_n)\right]\right) \tag{20}$$

where $K(\boldsymbol{x}_i) = (\kappa(\boldsymbol{a}_1, \boldsymbol{x}_i) - \mu_1, \cdots, \kappa(\boldsymbol{a}_s, \boldsymbol{x}_i) - \mu_s)^T$, $\mu_j = \sum_{i=1}^{n}\kappa(\boldsymbol{a}_j, \boldsymbol{x}_i)/n$ and $W = [\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_m] \in \mathbb{R}^{s \times m}$. It is interesting to find out that this embedding can be interpreted as first nonlinearly transforming each $\boldsymbol{x}_i \in \mathbb{R}^d$ to $K(\boldsymbol{x}_i) \in \mathbb{R}^s$ and then linearly projecting $K(\boldsymbol{x}_i)$ on W. Therefore, after transforming each $\boldsymbol{x}_i$ to $K(\boldsymbol{x}_i)$, the learning method of TPH can be used to learn W. As a result, more effective embeddings can be obtained in a relatively straightforward manner by introducing kernels. We leave this for our future work.

Another aspect of TPH that can be improved is the relative proximities between neighbors of data points preserved. In this paper, the relative proximities are revealed by distances, which are neighborhood rankings. In some nonlinear dimensionality reduction methods, such as locally linear embedding (LLE) [21], the relative proximities are revealed by local angles between neighbors and data points. To some extent, the preservations of local angle and distance are related and can be interpreted as two different ways to preserve local products [11]. In the next step, we would like to investigate whether the preservation of local angle can be incorporated to develop more effective hashing method.

4. EXPERIMENTAL RESULTS

4.1 Experimental Setup

Our experiments are carried out on three noted datasets: MINST70K[3], CIFAR10[4] and ANN-SIFT1M. The MNIST70K consists of 70K 784-dimensional images, each of which is associated with a digit label from '0' to '9', and is split into a baseset (training set, 60K) and a query set (10K). The CIFAR10 consists of 60K 32×32 images (baseset 50K, query set 10K) which are manually labeled into 10 classes. In our experiments, the images are represented with 512-dimensional GIST descriptors [18]. The ANN-SIFT1M [5] consists of almost 1M 128-dimensional SIFT descriptors [15], and contains three subsets: learning set (100K), baseset (1M) and query set (10K). We randomly select 1,000 points from the query set as queries in our experiments on ANN-SIFT1M.

There are many hashing methods, and some of them are either supervised [24, 13] or semi-supervised [25, 26]. Since our TPH in this work is essentially unsupervised, for fair comparison, we select some representative unsupervised methods for comparison. The selected baseline methods are LSH [1], PCAH [28], ITQ [3], SPH [29], AGH [14], BRE [9] (we use the unsupervised version), KLSH [10] and SIKH [20]. Among these methods, PCAH, ITQ, SPH and AGH are designed to preserve the original neighborhood relationships in Hamming space, while LSH, BRE, KLSH and SIKH are designed to approximate the distance between data points in the original data space or kernel space with Hamming distance. The source codes generously provided by the authors are used in our experiments. As for the parameters of each algorithm, we use the recommended settings in their papers.

In BRE, KLSH and TPH, a subset of dataset for training is essential. We randomly sample 2,000 points from the basesets of MINST70K and CIFAR10 (200 for each class), and 3,000 points from the baseset of ANN-SIFT1M as the training sets, respectively. For fair comparison, we use the same training set for these three methods. The regularization weight, α, is not fully tuned and is simply set to 0.1 for MINST70K, 0.2 for CIFAR10, and 1.0 for ANN-SIFT1M.

[3]http://yann.lecun.com/exdb/mnist/
[4]http://www.cs.toronto.edu/∼kriz/cifar.html

To better validate our method, we also compare TPH with Semi-supervised Hashing (SSH) [26] and Product Quantization (PQ) [5]. As the quantization error of PQ is much smaller than most hashing methods [29, 3], the Euclidean neighbor search performance of PQ is often better than hashing based methods [5, 17]. However, as the search process of PQ is performed in the original data space, it often has a higher query time. Moreover, as PQ ignores the manifold structure of dataset, it is not effective for semantic neighbor search. By contrast, many hashing methods, e.g. AGH, SSH and our TPH, are more effective to capture semantic neighbor.

4.2 Evaluations for Similarity Search

We first evaluate TPH and TPH-R for similarity search. Since MINST70K is fully annotated, the neighbors of each query image are defined as those images with the same digit labels. For CIFAR10, the ground truth of each query is its top 5,000 Euclidean neighbors in the original data space [3, 20]. And for ANN-SIFT1M, a returned point is considered as a true neighbor if it lies in the top 1% points closest to the query, measured by the Euclidean distance [25]. In the learning stage, the topo-weight of data pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is simply set to the Euclidean distance, i.e. $\tau_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$, and the simi-weight is $w_{ij} = \exp\{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/\sigma^2\}$ [29], where σ is the kernel width and is set to the average pairwise distance of a sampled subset of each dataset.

Figure 2 gives the precision-recall curve on MINST70K using 32 bits binary codes. For clarity, the results are shown in two parts. From Fig. 2 it is clear that our TPH and its extension TPH-R perform best for similarity search on this dataset. More specifically, TPH outperforms most baseline methods except for ITQ (and AGH in low-recall case), and TPH-R gives the best search performance under most settings. Moreover, as shown on Fig. 2 and Fig. 3(c), even with a relatively short binary code (32 bits), the search performance of TPH-R is still better than all baseline methods with longer binary codes (96 bits). These comparisons indicate that, TPH preserves the neighborhood relationships better by learning from the topology of the intrinsic manifold structure of MINST70K. It is interesting to note that, in these experiments, the ground truth is defined based on semantic label. Although all the evaluated methods are unsupervised, they give reasonable good performance for semantic neighbor search, especially the TPH-R. This is because that, in MINST70K, mostly, the Euclidean distances between data points with same labels are smaller than the distances between data points with different labels. By using the Euclidean distance as topo-weight, neighbor pairs are given smaller weights than non-neighbor pairs, which is similar with adopting the label-based weighting scheme in Section 3.3. Therefore, the experimental results also indicate that TPH is capable of capturing semantic neighbors with the label-based weighting scheme.

Figure 3-5 give the precision-recall curves on MINST70K, CIFAR10 and ANN-SIFT1M under different code lengths, respectively. Once again, we can easily find out that our TPH and TPH-R outperform other state-of-the-art methods under most settings. These comparisons demonstrate the effectiveness of incorporating the neighborhood ranking information with binary codes learning. Note that, the performance improvement of TPH-R on ANN-SIFT1M is not as remarkable as those on MINST70K and CIFAR10. This result is reasonable, since the training samples is only 3‰ of the entire ANN-SIFT1M dataset. This sampling sparsity makes the data topology of the training set non-representative, thus the learned W are not optimal. To verify this, we retrain our model with 30K training samples and report the mAP (21) of TPH-R: The mAP values of TPH-R with 32, 48, 64 and 96 bits codes are 0.3327, 0.4015, 0.5022 and 0.5301, respectively. In this case, TPH-R outperforms ITQ under all settings (mAPs of ITQ are given in Table 1), which also indicates our method become more effective with more training samples.

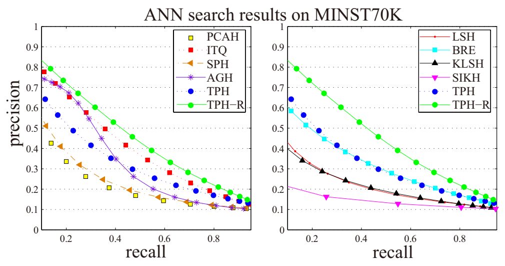


Figure 2: Precision-recall curve on MINST70K. All methods use 32 bits code.

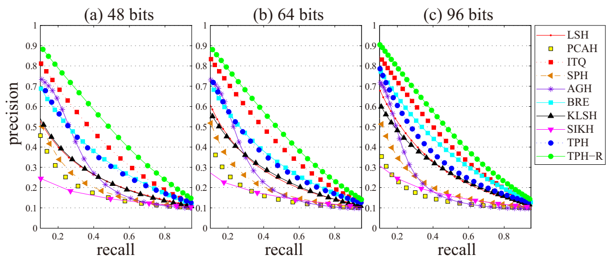


Figure 3: Precision-recall curves on MINST70K using different code lengths: (a) 48; (b) 64; (c) 96.

4.3 Evaluations for Topology Preserving

The above experiments demonstrate the superiority of TPH for similarity search. In this section, we conduct a series of experiments to evaluate the efficacy of TPH for the preservation of relative proximities, which is the ranking of neighbors in this work. In these experiments, the mAP is used as the performance metric. It is defined as:

$$\begin{aligned} \text{Precision} &= \frac{\text{number of retrieved neighbors}}{\text{number of all retrieved points}} \\ \text{mAP} &= \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{n_i}\sum_{k=1}^{n_i}\text{Precision}(R_{ik}) \qquad (21) \end{aligned}$$

where Q is query set, $|Q|$ is its cardinality and n_i is the number of neighbors of query $\boldsymbol{q}_i \in Q$ in dataset. The neighbors are ordered as $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{n_i}$ and R_{ik} is the set of ranked retrieval results from the top result to $\boldsymbol{x}_k$. This definition says that, for two hashing methods returning the same number of neighbors for a query, if one gives a better ranking for the neighbors, it would have a higher mAP, which indicates it preserves the local data topology better.

The mAP values of different methods with different code lengths on MINST70K, CIFAR10, and ANN-SIFT1M are given in Table 1. It is obvious that our TPH outperforms almost all baseline methods except for ITQ, and its exten-

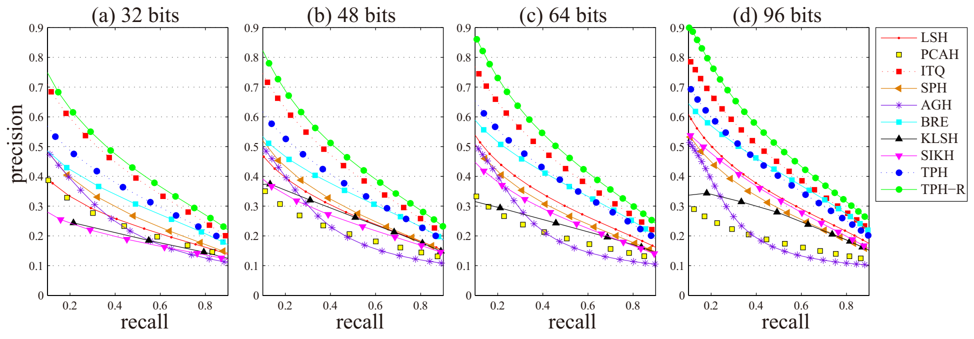


Figure 4: Precision-recall curves on CIFAR10 dataset. Code lengths: (a) 32; (b) 48; (c) 64; (d) 96.

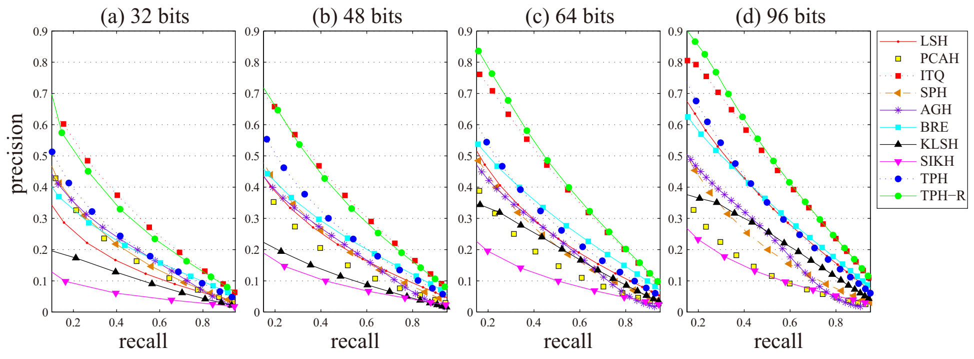


Figure 5: Precision-recall curves on ANN-SIFT1M dataset. Code lengths: (a) 32; (b) 48; (c) 64; (d) 96.

sion TPH-R again gives the best performance under most settings. These experimental results demonstrate that TPH and TPH-R preserve the local topology of a dataset better.

4.4 Leveraging Label Information

In this section, we show the performance of semantic neighbor search of TPH by leveraging label information. With label information, Topo-Training Matrix $\mathit{\Gamma}_t$ and Simi-Training Matrix $\mathit{\Gamma}_s$ can be constructed to better capture the semantic structure of a dataset, extending TPH semi-supervised.

Our experiments are carried out on MINST70K and CIFAR10. Each image in MINST70K is labeled with a digital number from '0' to '9' and each image in CIFAR10 is labeled into one of the following ten classes: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*. The neighborhood relationships are defined based on semantic labels. We adopt the label-based weighting scheme in Section 3.3 such that for a neighbor pair, its topo-weight is set to a small value, while for a non-neighbor pair, its topo-weight is set to a relatively large value. In the construction of $\mathit{\Gamma}_t$, for a neighbor pair, we set $\tau_{ij} = 1.0$, and for a non-neighbor pair, we set $\tau_{ij} = 10.0$ to ensure that the corresponding binary codes can be well separated in Hamming space. In the construction of $\mathit{\Gamma}_s$, the simi-weight is $w_{ij} = 1.0$ for a neighbor pair and $w_{ij} = \exp\{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/\sigma^2\}$ for a non-neighbor pair. For comparison, we select some methods which give good performances in our previous experiments. In addition, we also compare our method with PQ [5] and SSH [26] for better validation. For PQ, the original data is decomposed into 8 subspaces and the asymmetric distance calculation (ADC) is used for neighbor search. For SSH, we use the same training set as that of TPH.

Table 2 reports the mAP values of different methods for semantic neighbor search on MINST70K and CIFAR10, and Fig. 6 gives the qualitative retrieval results of TPH leveraging label information applied on CIFAR10 using 32 bits codes. Our TPH with label information is denoted as **TPH-L**. We can easily see that, when label information is incorporated, the retrieval results are much more semantically consistent. Moreover, TPH-L gives a better ranking for the neighbors of each query. Since most of the compared baselines are unsupervised, the comparison results given here are not to show that TPH outperforms these methods for semantic neighbor search, but to demonstrate that, by leveraging label information, our TPH can be easily extended to a semi-supervised method to capture semantic neighbor effectively.

4.5 Summary

In summary, the experimental results demonstrate the efficacy of TPH for fast similarity search. From the results we can conclude that, by learning from the local topology of the intrinsic manifold structure of a dataset, TPH can generate more optimal Hamming embeddings, which not only preserve the neighborhood relationships between data points, but also preserve the neighborhood ranking of each data point. This finding has important implications for learning to hashing: the neighborhood ranking information is also valuable for effective hashing. Moreover, by leveraging label information, the learning stage of TPH can be easily extended to a semi-supervised manner, which gives TPH the capability of capturing semantic neighbors. In the future, we will refine the formulation of TPH, making the label information be better incorporated with the learning stage.

Table 1: mAP values on MINST70K, CIFAR10 and ANN-SIFT1M under different settings. The best mAP among TPH, TPH-R and other methods is shown in bold face.

dataset	MINST70K				CIFAR10				ANN-SIFT1M			
method \ #bits	32	48	64	96	32	48	64	96	32	48	64	96
PCAH	0.2456	0.2272	0.2159	0.2009	0.2409	0.2261	0.2155	0.2000	0.2136	0.2286	0.2167	0.1931
ITQ	0.4480	0.4620	0.4752	0.4815	0.4362	0.4591	0.4724	0.4893	**0.3173**	**0.3820**	0.4665	0.5083
SPH	0.2699	0.2544	0.2569	0.2604	0.2770	0.3023	0.2950	0.3131	0.2345	0.2750	0.3231	0.3103
AGH	0.3674	0.3311	0.3074	0.2777	0.2540	0.2515	0.2467	0.2390	0.2373	0.2835	0.3312	0.3390
LSH	0.2561	0.3064	0.3328	0.3702	0.2667	0.3111	0.3462	0.3807	0.1696	0.2543	0.3231	0.3603
BRE	0.3604	0.4006	0.4014	0.4470	0.3143	0.3484	0.3824	0.4219	0.2432	0.2998	0.3430	0.3553
KLSH	0.2227	0.2848	0.2956	0.3207	0.1996	0.2670	0.2100	0.2392	0.1045	0.1364	0.2367	0.3243
SIKH	0.1454	0.1562	0.1717	0.1867	0.1922	0.2481	0.2709	0.3309	0.0610	0.0960	0.1171	0.1302
TPH	0.3648	0.3878	0.3864	0.3934	0.3878	0.4144	0.4346	0.4625	0.2766	0.3312	0.3648	0.3687
TPH-R	**0.5027**	**0.5194**	**0.5213**	**0.5283**	**0.4558**	**0.4707**	**0.4936**	**0.5160**	0.3056	0.3810	**0.4803**	**0.5210**

Table 2: mAP values of semantic neighbor search on MINST70K and CIFAR10.

MINST70K

#bits	PQ	SSH	TPH-R	TPH-L
32	0.4454	0.4670	0.5027	**0.5161**
64	0.4521	0.4861	0.5213	**0.5333**

CIFAR10

#bits	ITQ	SPH	AGH	BRE
32	0.1634	0.1307	0.1340	0.1493
64	0.1740	0.1313	0.1345	0.1578
	PQ	SSH	TPH-R	TPH-L
32	0.1766	0.1982	0.1658	**0.2126**
64	0.1820	0.2144	0.1772	**0.2223**

5. CONCLUSIONS

To capture meaningful neighbors, most hashing methods are developed to preserve the neighbor relationships while ignoring the neighborhood rankings. In this paper, we show that the neighborhood rankings are as important as the neighborhood relationships for learning to hashing, and propose a novel hashing method, Topology Preserving Hashing (TPH), by incorporating the neighborhood ranking information with hash function learning. Our approach is distinct from prior works by not only preserving the neighborhood relationships between data points, but also preserving the neighborhood rankings. Since the ranking information is encoded in the codes, the ranking ambiguity of most hashing methods is also well alleviated. Experimental results on three large-scale image datasets containing up to one million high-dimensional data points show the efficacy of TPH. Moreover, although our method is entirely unsupervised in this work, it can be easily extended to a semi-supervised method for semantic neighbor search as demonstrated. In the future, we will evaluate more sophisticated techniques to solve the problem of topology preserving and apply our method to a large spectrum of information retrieval problems such as image retrieve and near-duplicate detection.

6. ACKNOWLEDGMENTS

This work is supported by National Nature Science Foundation of China (61273247, 61271428), National Key Technology Research and Development Program of China (2012-BAH39B02). This work was supported in part to Dr. Qi Tian by ARO grant W911BF-12-1-0057, NSF IIS 1052851, Faculty Research Awards by Google, FXPAL, and NEC Laboratories of America, and 2012 UTSA START-R Research Award respectively. This work was supported in part by NSFC 61128007.

7. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006.

[2] J. L. Bentley and B. Labo. K-d trees for semidynamic point sets. In *SOCG*, pages 187–197, 1990.

[3] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2012.

[4] M. Jain, H. Jégou, P. Gros, et al. Asymmetric hamming embedding: taking the best of our bits for large scale image search. In *ACM Multimedia*, 2011.

[5] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33:117–128, 2011.

[6] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian. Super-bit locality-sensitive hashing. In *NIPS*, pages 108–116, 2012.

[7] Y.-G. Jiang, J. Wang, X. Xue, and S.-F. Chang. Query-adaptive image search with hash codes. *IEEE Trans. Multimedia*, 15(2):442–453, 2013.

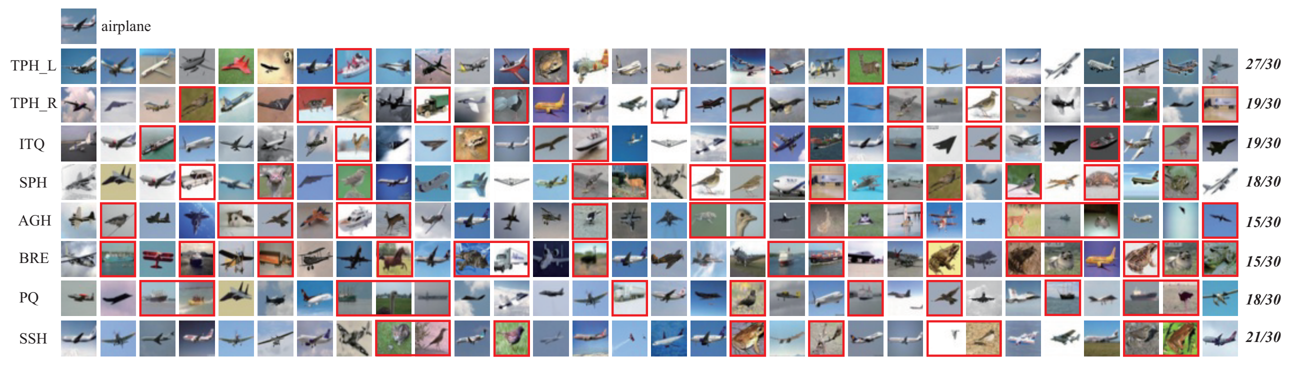
[8] A. Joly and O. Buisson. A posteriori multi-probe locality sensitive hashing. In *ACM Multimedia*, 2008.

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

[10] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34:1092–1104, 2012.

[11] J. Lee and M. Verleysen. *Nonlinear dimensionality reduction.* Springer, 2007.

[12] P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu. Spectral hashing with semantically consistent graph for image



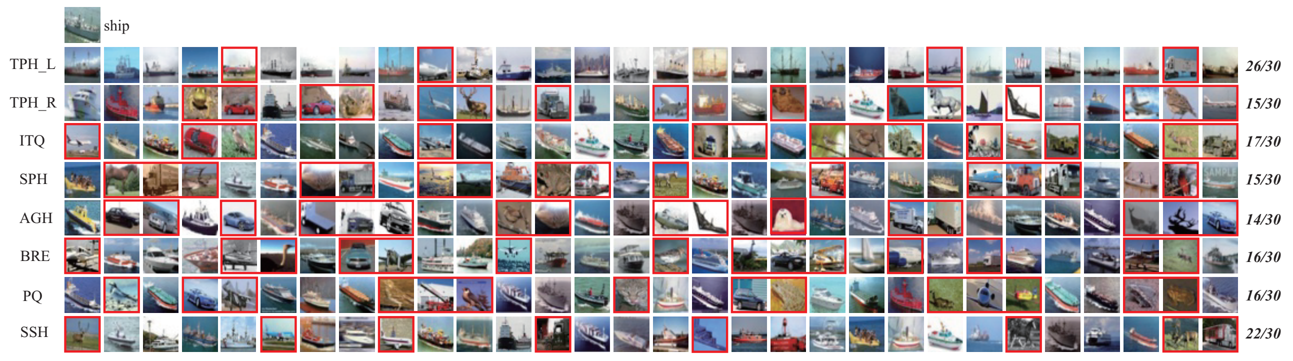


Figure 6: Qualitative retrieval results on CIFAR10 for two queries. Apparently, the retrieval quality of TPH with label information (the first row) is much better. The semantical consistency and ranking of the retrieval results are both significantly improved. Red rectangle denotes false positive. Best viewed in color.

indexing. *IEEE Trans. Multimedia*, 15(1):141–152, 2013.

[13] W. Liu, J. Wang, R. Ji, Y. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[14] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[16] D. Luo, C. Ding, F. Nie, and H. Huang. Cauchy graph embedding. In *ICML*, pages 553–560, 2011.

[17] M. Norouzi and D. Fleet. Cartesian k-means. In *CVPR*, pages 3017–3024, 2013.

[18] A. Oliva and A. B. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001.

[19] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[20] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009.

[21] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[22] R. Salakhutdinov and G. E. Hinton. Semantic hashing. *Int. J. Approx. Reason*, 50:969–978, 2009.

[23] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM Multimedia*, pages 423–432, 2011.

[24] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: Improved matching with smaller descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(1):66–78, 2012.

[25] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *ICML*, pages 1127–1134, 2010.

[26] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, 2012.

[27] M. Wang, X.-S. Hua, J. Tang, and R. Hong. Beyond distance measurement: constructing neighborhood similarity for video annotation. *IEEE Trans. Multimedia*, 11(3):465–476, 2009.

[28] X. Wang, L. Zhang, F. Jing, and W. Ma. AnnoSearch: image auto-annotation by search. In *CVPR*, pages 1483–1490, 2006.

[29] Y. Weiss, A. B. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[30] L. Zhang, Y. Zhang, J. Tang, K. Lu, and Q. Tian. Binary code ranking with weighted hamming distance. In *CVPR*, pages 1586–1593, 2013.

[31] W. Zhou, Y. Lu, H. Li, Y. Song, and Q. Tian. Spatial coding for large scale partial-duplicate web image search. In *ACM Multimedia*, pages 511–520, 2010.

[32] W. Zhou, Y. Lu, H. Li, and Q. Tian. Scalar quantization for large scale image search. In *ACM Multimedia*, pages 169–178, 2012.