Hashing with non-linear manifold learning

Yanzhen Liu*, Xiao Bai*, Cheng Yan*, Jing Wang† and Jun Zhou†
*School of Computer Science and Technology
Beihang University
Email: lyzeva,baixiao,beihangyc@buaa.edu.cn
†School of Information and Communication Technology
Griffith University
Email: jing.wang5,jun.zhou@griffith.edu.au

***Abstract*—The amount of data is exploding with the development of Internet and multimedia technology. Rapid retrieval of mass data is becoming more and more important. To meet the demand of the rapid retrieval, many approximate nearest neighbor methods have been proposed to accelerate the exhaustive search process. Hashing is such an example with great balance of time and accuracy. Hashing methods achieve quick retrieval by converting the high-dimensional raw data into a binary hash code, keeping the similarity of original data in mapped hash codes. Many hashing approaches use the Euclidean distance as similarity measurement. However, data in many datasets are distributed on a non-linear manifold, such that geodesic distance on manifold can represents the semantic similarity of original data points more accurately than the Euclidean distance. This enables better preservation of the sematic similarity in the hash code when mapping the original dataset to low-dimensional space. In this paper, we propose to use Isometric Mapping for dimensional reduction and utilize iterative quantization to reduce quantization loss during hashing process. The experiments show that our manifold learning method outperforms several alternative hashing methods. The retrieval performance is further boosted after iterative quantization process is added to the Diffusion Hashing and Spectral Hashing.**

I. Introduction

Production and communication of data become easier with the development of mobile Internet and multimedia technology. This leaves mass data to be dealt with in daily information retrieval tasks. Nearest Neighbor (NN) search is an important solution to find the most similar data point of a query in the database. Unfortunately, for mass data, particularly in high-dimensional image sets, the cost of linear scan is too high and makes retrieval process very slow. This leads to the wide adoption of sublinear Approximate Nearest Neighbor (ANN) methods. Hashing is an important class of ANN methods, which tries to map the raw dataset into binary hash codes and makes the hamming distance between the hash codes be positive correlated to the degree of similarity between raw data in the dataset. With hash codes, rapid ANN retrieval can be realized by comparing the hamming distances of binary hash codes. Many hashing schemes have been proposed, and can be divided into unsupervised hashing methods [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11] and supervised hashing methods [12], [13], [14], [15], [16] depending on whether supervised information is involved.

The aim of hashing is to find a mapping from data points to binary codes, preserving the similarity relation of orignal data points in mapped hashing codes. Thus the essence of hashing is to construct a mapping from Euclidean space to hamming space. Generally, the dimension of data points is more than the length of hash codes. The process of hashing can be separated into dimensionality reduction and 0-1 quantization. To get better hash codes, the dimensionality reduction process should preserve the similarity relationship of the dataset and the 0-1 quantization process should bring information loss as less as possible. To reduce quantization loss, [17] proposed an iterative quantization approach (ITQ). ITQ tries to rotate the data after dimensionality reduction. It calculates an rotation matrix that makes the quantization loss least. Lots of experiments show that ITQ works well in most situations.

Unsupervised datasets have no information about true classes, so many unsupervised methods directly adopt Euclidean distance as similarity measure. For many image datasets, data points may distribute on a manifold. In these cases, geodesic distance on the manifold represents the semantic similarity of original data points more accurately than the Euclidean distance. In other words, the most similar data pairs may not be the nearest neighbor in Euclidean space, but they are the nearest neighbor on the same manifold. When mapping the dataset into a low-dimensional space, to achieve similar hash codes with high sematic similarity, the manifold structures of datasets shall be considered.

For linear manifold, linear methods such as LSH [18] and PCA-ITQ [17], [19] are applicable. But for non-linear manifold such as Swiss Roll, further exploration is needed to catch the manifold structure in hash codes. Many non-linear manifold learning methods are motivated by graph method which realizes low-dimension embedding of data with manifold distribution in the Euclidean space. These methods can be divided into two categories, global preserving manifold learning method and local preserving manifold learning method. Isometric Mapping (ISOMAP) [20] and Maximum Variance Unfolding (MVU) [21] are global preserving manifold learning methods. Laplace Eigenmaps (LE) [22], Locally Linear Embedding (LLE) [23], Hessian Locally Linear Embedding (HLLE) [24] and Diffusion Map (DFM) [25], [26] are local preserving manifold learning methods. Many hashing methods utilize those manifold learning methods in their dimension reduction process. Works in this direction include Spectral Hashing (SH) [27], Anchor Graph Hashing

(AGH) [2], Diffusion Hashing (DH) [28], Inductive Manifold Hashing (IMH) [29] and Locally Linear Hashing (LLH) [30].

Our paper aims at improving the ability of maintaining semantic similarity of data after hashing. We improve the accuracy of retrieval by utilizing manifold learning method for dimensionality reduction and using iterative quantization to reduce the quantization loss during binarization. In this paper, we concentrate our attention on the quantization process of manifold-based hashing schemes. For some of the existing manifold-based hashing schemes [27], [28], their quantization process is very simple only with a sign function, so we minimize the quantization loss with iterative quantization (ITQ) [17], [19] to raise the performance. And we do more experiments to find how ITQ affects these manifold-based hashing schemes.

The remaining content of this paper is arranged as follows. Section II reviews some related work of manifold-based hashing methods. Secion III describes the proposed hashing methods in details. First, we introduce a dimensionality reduction process (Section III-A) which is implemented via three methods, i.e., ISOMAP (Section III-A1), Laplace Eigenmaps (Section III-A2) and Diffusion Maps (Section III-A3). Then we describe the quantization process in Section III-B and out-of-sample extension in Section III-C. The experiments to evaluate our method are given in Section IV, followed by conclusions in Section V.

II. Related Work

In high dimensional space, data normally lies on a manifold. Linear dimensionality reduction methods perform well on simple linear manifold, but is not capable of dealing with non-linear manifold with complex structure. In order to preserve the semantic relevance of original data, manifold learning are introduced into hashing methods. Spectral Hashing(SH) [27] and Diffusion Hashing (DH) [28] are representative frameworks.

Laplace Eigenmaps (LE) [22] is an manifold learning method early-used in hashing. If we regard the dataset as a graph, whose nodes are datapoints and whose edges are the distance between datapoints. And the object formulation of LE is equivalent to balanced graph partition problem: minimizing $cut(A, B)$ with the requirement $|A| = |B|$. A classical work is Spectral Hashing (SH) [27]. The object formulation of SH is the same as that of LE, and SH adds additional constraints of hash codes' efficiency. But SH assumes that the dataset has uniform distribution, and the accuracy of SH is affected. Anchor Graph Hashing (AGH) [2] is another LE based hashing scheme. AGH no longer needs the uniform distribution assumption and uses anchor points to reduce the time complexity of the algorithm.

Diffusion Hashing (DH) [28] is based on Diffusion Maps (DFM) [25], [26]. DM has the same objective formulation of LE, but the kernel matrix is no longer Gaussian Kernel but the anisotropic diffusion kernel, better dealing with non-linear behavior of dataset in higer dimension. And DH gives an linear transformation to original data before the dimension reduction process, thus DH can easily get binary code without considering out-of-sample extension problem.

III. Hashing Methods with Non-linear Manifold Learning

Given a set of data points $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]^\top$ $\left(\mathbf{x}_k \in \mathbb{R}^d, X \in \mathbb{R}^{n\times d}\right)$, we want to get a mapping from X to a binary hash code set $B \in \{-1, 1\}^{n\times c}$ that the hamming distance in B best approximates the data similarity in X.

We explore the geodesic distance on manifold rather than the Euclidean distance as the quantitative measurement of data similarity. Therefore, in the mapping process, we choose to preserve geodesic distance between data points in X and then use a non-linear dimensionality reduction method. After dimensionality reduction with the manifold learning method, we get the low-dimensional embedding $Y \in \mathbb{R}^{n\times c}$, and we quantize Y into a binary hash code B.

Here is a list of mathematical symbols appearing in the following sections:

b: number of bits of the final hash code.
ξ: eigenvectors of the kernel matrix.
λ: eigenvalue of the kernel matrix
$\mathbf{y}_k$: embedding vector of original data vector $\mathbf{x}_k$.
Y: embedded data matrix, $Y = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_b)^\top$.
B: binary code matrix.

A. Dimensionality Reduction

1) Isometric Mapping Hashing with ISOMAP: Isomap is a non-linear dimensionality reduction method which aims at preserving the geodesic distance of the manifold structure. Real geodesic distance is difficult to compute, so Isomap takes the shortest path distance to approximate the geodesic distance in large dataset.

We can construct a neighborhood graph G with the dataset X, whose edges connect only neighboring nodes with the Euclidean distance of neighboring nodes as the weights. To properly define the neighborhood, we set a threshold ϵ. Only node pairs whose Euclidean distances are below the threshold are considered as neighboring nodes. Alternatively, we can choose a number k. Only node pairs being k-nearest-neighbor to each other are regarded as neighboring. With this neighborhood graph G, we compute the shortest path distance between all node pairs in G to approximate the geodesic distance of those node pairs. We denote the approximate geodesic distance matrix as D.

For ISOMAP, we should apply Multidimensional Scaling (MDS) [31] first to the landmark points. We can construct matrix

$$K_s = -\frac{1}{2}HDH, \tag{1}$$

where $H = E - \frac{1}{t}ee^T$ (e is a column vector). We can get the eigenvectors $\xi_k\,(1 \leqslant k \leqslant b)$ of the top b eigenvalues $\lambda_k\,(1 \leqslant k \leqslant b)$ of K. Then the low-dimensional embedding can be calculated as

$$Y = \Lambda^{\frac{1}{2}}V. \tag{2}$$

2) Spectral Hashing with Laplace Eigenmap: The basic idea of Laplace Eigenmaps (LE) is making the points close to each other in the dataset still close to each other in low-dimension space. LE reconstructs the local structure of the Manifold by constructing the similar diagram. The objective function of Laplace Eigenmaps is as follows

$$\min \sum_{i,j} W_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|^2, \tag{3}$$

where $W_{ij} = exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\epsilon^2)$ is a Gaussian kernel. After linear algebraic transformation, the optimization problem above can be expressed in the form of a matrix vector:

$$\min tr(X^\top L X), \ s.t. X^\top D X = I. \tag{4}$$

L is a Laplacian matrix $L = D - W$, W is a Gaussian kernel matrix and D is a diagonal matrix $D_{ii} = \sum_j W_{ij}$.

Calculate the eigenvectors ξ and eigenvalues λ of L:

$$L\xi = \lambda D \xi. \tag{5}$$

We use the eigenvectors corresponding to smallest b nonzero eigenvalues as the output of dimensionality reduction. That is

$$Y = [\xi_1, \xi_2, ..., \xi_b]. \tag{6}$$

3) Diffusion Hashing with Diffusion Maps: Diffusion Map aims at representing manifold efficiently. It utilizes an anisotropic diffusion kernel to better represent manifold structure in high dimensional space.

With the training data $X = \{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{n \times d}$, we compute the mean of $\mathbf{x}$

$$\bar{\mathbf{x}} = 1/n \Sigma_i \mathbf{x}_i. \tag{7}$$

Then compute Gaussian kernel W between each data

$$W_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2). \tag{8}$$

We utilize an anisotropic diffusion kernel to normalize W with the diagonal matrix Q whose elements are the sum of each column

$$K = Q^{-1} W Q^{-1}, Q_{ii} = \sum_j W_{ij}. \tag{9}$$

The diagonal matrix Q_{ii} has a characteristic by which the element becomes smaller when the data distribution is sparse, whereas it becomes larger when the distribution is dense. By normalizing the data with the reciprocal of these elements in Q_{ii} , we expect to cope with a variety of data distributions. It is also possible to obtain the transition probability matrix P from matrix K, if we normalize it such that the sum of each column becomes

$$P = D^{-1} K, D_{ii} = \sum_j K_{ij}. \tag{10}$$

With transition probability matrix P, we get the objective formulation

$$\arg\min \sum_{i,j} (\mathbf{y}_i - \mathbf{y}_j)^2 P_{ij}, \tag{11}$$

And it can be transformed into a generalized eigenvalue problem

$$X^\top P X \xi = \lambda X^\top X \xi, \tag{12}$$

from which we get eigenvectors ξ_i computed.

Because $X^\top X$ is not always non-singular, we first do singular value decomposition on X: $X = U \Sigma V^\top$. Denote r as the rank of matrix X, then Σ is an $r \times r$ diagonal matrix, and U and V are orthogonal $n \times r$ and $d \times r$ matrices . Let

$$\tilde{X} = \Sigma V = U \Sigma, \tag{13}$$

then the generalized eigenvalue problem in Equation (12) is equivalent to

$$\tilde{X}^\top P \tilde{X} \tilde{\xi} = \lambda \tilde{X}^\top \tilde{X} \tilde{\xi}, \tag{14}$$

with

$$\xi = V \tilde{\xi}. \tag{15}$$

After the generalized eigen decomposition process of Equation (14), we choose eigenvectors $\tilde{\xi_k} (1 \le k \le b)$ of the top b eigenvalues and transfer $\tilde{\xi_k}$ to ξ_k by Equation (15). And we get a projection matrix by the training process

$$P_r = [\xi_1, \xi_2, ..., \xi_b]. \tag{16}$$

Then the low-dimensional embedding can be calculated as

$$Y = P_r^\top (X - \bar{X}) \tag{17}$$

where $\bar{X}$ is defined on n $\bar{\mathbf{x}}$ in a column-wise style $[\bar{\mathbf{x}}, \bar{\mathbf{x}}, ..., \bar{\mathbf{x}}]^\top$.

B. Quantize Low-dimension Embedding with ITQ

After getting the low-dimension embedding Y, we are able to quantize it into binary code. In order to get the binary code, we need to quantize each dimension in the low-dimension embedding into 0 and 1 according to the formula

$$B = sign(Y). \tag{18}$$

Direct quantization may cause great loss as explained in [17], [19]. Therefore, we try to calculate an optimized rotation matrix to minimize the quantization loss Q.

$$Q = \|B - YR\|_F^2. \tag{19}$$

To solve this optimization problem, an iterative method was proposed to solve B and R one after the other [17], [19]. When solving B, we fix R and calculate B with sign function $B = sign(YR)$. When solving R, we fix B and the problem becomes the classic Orthogonal Procrustes Problem. It can be solved by computing the Singular Value Decomposition (SVD) of $B^\top V$ as $S \Omega \hat{S}$, and we get $R = \hat{S} S^\top$. Finally, we get a local minimum of the quantization loss Q and its corresponding optimal rotation matrix R. Then binary hash code of training set can be represented as $B = sign(YR)$.

C. Out-of-sample Extension

By applying the described method above on the training set, we can get the binary codes of the whole training set. If we get another single testing data, how can we get its binary code? For Diffusion Hashing (DH), there is an projection matrix training process and an out-of-sample can easily be dealt with by Equation (17). For methods like SH and IsoMH, we directly get the hash code from the training process and there is no specific equation for an out-of-sample. We can add the out-of-sample to the training set and run the training algorithm again for the new hash code. We also can directly do l_2 scan because it is quicker than the out-of-sample training process. To solve this problem, we adopt an out-of-sample extension methods to calculate the approximate ISOMAP embedding.

1) Out-of-sample Extension of IsoMH: There is a solution of out-of-sample extension of ISOMAP solved by [32]. We utilize this solution in our IsoMH.

Given a query $\mathbf{q}$, we first need to calculate its approximate geodesic distance, in other words, its shotrtest path distance to all other training set points. There is no need to run Dijkstra one more time. If the neighborhood of $\mathbf{q}$ is $\mathbf{x}_{l_1}, \mathbf{x}_{l_2}, ..., \mathbf{x}_{l_k}$ $(1 \leqslant l_i \leqslant n)$, then the shortest path distance of $\mathbf{q}$ to all other point in X can be calculated by the shortest path distance of its neighbors. If we denote the shortest path distance from $\mathbf{x}_i$ to $\mathbf{x}_j$ by $D(\mathbf{x}_i, \mathbf{x}_j)$, it is intuitive that

$$D(\mathbf{x}, \mathbf{q}) = \min_i \{\|\mathbf{q} - \mathbf{x}_{l_i}\| + D(\mathbf{x}_{l_i}, \mathbf{x})\}. \tag{20}$$

With the newly calculated $D(\mathbf{x}, \mathbf{q})$, we can calculate the embedding $\mathbf{y}$ of an out-of-sample $\mathbf{x}$ according to the expression

$$\mathbf{y}_k(\mathbf{q}) = \frac{1}{2}\frac{1}{\sqrt{\lambda_k}}\sum_{i=1}^{n} v_{ki}\left(E_{\mathbf{x}'}\left[D^2\left(\mathbf{x}', \mathbf{x}_i\right)\right] - D^2(\mathbf{x}_i, \mathbf{q})\right), \tag{21}$$

given in [32]. And it can be represented in matrix as

$$\mathbf{y} = \frac{1}{2}\Lambda^{\frac{1}{2}} V \eta, \tag{22}$$

where $\eta = E_{\mathbf{x}'}\left[D^2\left(\mathbf{x}', \cdot\right)\right] - D^2(\mathbf{q}, \cdot)$ is a column vector.

2) Out-of-sample Extension of SH: The out-of-sample extension has been given in Spectral Hashing [27]. And the author assumes that the data is in multidimensional uniform distribution. Then we get the eigenfunction $\Phi_k(\mathbf{x})(1 \leq k \leq b)$:

$$\Phi_k(\mathbf{x}) = sin(\frac{\pi}{2} + \frac{k\pi}{b_k - a_k}\mathbf{x}), \tag{23}$$

where the range of this multidimensional distribution in the kth dimension is $[a_k, b_k]$. So the hash code of an out-of-sample $\mathbf{q}$ is

$$\mathbf{b}_q = [sign(\Phi_1(\mathbf{q})), sign(\Phi_2(\mathbf{q})), ..., sign(\Phi_b(\mathbf{q}))]^\top. \tag{24}$$

We summarize the whole algorithms of IsoMH, SH-ITQ and DH-ITQ by pseudocodes in Algorithm 1, Algorithm 2 and algorithm 3.

Algorithm 1: Pseudocode of Isometric Mapping Hashing

Data: training data X, code length b, out-of-sample $\mathbf{q}$
Result: hash code matrix of the dataset B, hash code of out-of-sample data $\mathbf{b}_q$
Compute Euclidean distances of all data points;
Sort Euclidean distances to identify neighborhoods;
Construct adjacency matrix A of neighborhood graph G;
Calculate shortest path distance D of graph G;
Execute MDS on D and get V, Λ and Y;
Execute ITQ on Y to get the best quantization code B, with rotation matrix R;
Compute Euclidean distances of $\mathbf{x}_q$ to all training data;
Decide the k neighborhood point set of $\mathbf{q}$: $\mathbf{x}_{l_1}, \mathbf{x}_{l_1}, ..., \mathbf{x}_{l_k}$;
Calculate shortest path distance $\mathbf{d}_q$ from $\mathbf{q}$ to all other training data according to Equation (20);
Calculate Euclidean embedding $\mathbf{y}_q$ of out-of-sample point according to Equation (21);
$\mathbf{b}_q = sign(R^\top \mathbf{y}_q)$;

Algorithm 2: Pseudocode of Spectral Hashing with ITQ

Data: training data X, code length b, out-of-sample $\mathbf{q}$
Result: hash code B,hash code of out-of-sample data $\mathbf{b}_q$
Excute PCA on X and reduce the dimension of X to $n * b$;
Compute Gaussian Kernel K, Laplace Matrix L;
Execute Eigendecompostion on $L - K$ and get V, Λ and Y;
Execute ITQ on Y to get the best quantization code B, with rotation matrix R;
Calculate $\phi(\mathbf{x})$ by Equation (23);
$\mathbf{b}_q = sign([\phi_1(\mathbf{q}), \phi_2(\mathbf{q}), ..., \phi_b(\mathbf{q})]R)^\top$;

Algorithm 3: Pseudocode of Diffusion Hashing with ITQ

Data: training data $X \in \mathbb{R}^{n \times d}$, code length b
Result: projection matrix $P_r = \{\xi_i\}_{i=1}^n \in \mathbb{R}^{d \times b}$, rotation matrix $R \in \mathbb{R}^{b \times b}$,hash code of the training set B
Compute mean of $\mathbf{x}$: $\bar{\mathbf{x}}$; Compute Gaussian kernel matrix W;
Compute the anisotropic diffusion kernel K;
Compute the transition probability matrix P;
Do SVD on the data matrix X: $X = U\Sigma V^\top$ and get $\tilde{X} = \Sigma V = U\Sigma$;
Solve the generalized eigen-decomposition problem $\tilde{X}^\top P \tilde{X}\tilde{\xi} = \lambda \tilde{X}^\top \tilde{X}\tilde{\xi}$ and get the eigenvectors $\tilde{\xi}_1, \tilde{\xi}_2, ..., \tilde{\xi}_b$ with the largest b eigenvalues, making the matrix $\tilde{P}_r$;
The real projection matrix $P_r = V\tilde{P}_r$;
Compute the projected data $Y = XP_r$;
Execute ITQ on Y and get the rotation matrix R;
$B = sign(YR)$;

IV. EXPERIMENTS

To test the effectiveness of our method, we conduct experiments on 2 datasets: **MNIST** [33] and **CIFAR-10** [34]. For comparison, we also run several state-of-the-art methods which include Locality-Sensitive Hashing(LSH) [18], Spectral Hashing(SH) [27], Diffusion Hashing(DH) [28], Iterative Quantization(PCA-ITQ) [17], [19]. SH and DH are manifold-based hashing schemes while LSH and PCA-ITQ are linear hashing scheme. LSH, PCA-ITQ and DH learn a linear projection matrix while IsoMH and SH learn only hash codes and get out-of-sample by other methods. For each method, we test with hash codes of 12, 16, 32, 48 and 64 bits.

In our experiment, we first randomly choose data points from the dataset in an appropriate amount, then separate them into a training set and a testing set. The training set is used to learn parameters in the hash functions and build the hash code table. The testing set is used to calculate hash codes and retrieve in the hash code table. Training set need to be large enough to describe the distribution of the whole dataset. In this paper, we randomly choose 10000 points from each database, allocating 9000 points to the training set and 1000 to the testing set.

We evaluate each method using four criteria which are precision, recall, mean average precision, training and test time. Precision is denoted as p and is defined by

$$p = \frac{Number\ of\ retrieved\ true\ neighbor\ pairs}{Number\ of\ retrieved\ neighbor\ pairs}. \quad (25)$$

Recall is denoted as r and is defined by

$$r = \frac{Number\ of\ retrieved\ true\ neighbor\ pairs}{Total\ number\ of\ true\ neighbor\ pairs}. \quad (26)$$

Mean average precision is denoted by MAP and is defined by the equation

$$MAP = \int_0^1 p(r)dr. \quad (27)$$

We use the hamming distances between hash codes as the threshold to judge whether the query in the testing set is in the same class as a specific point in the training set. We choose different values of threshold and make the precision curve, recall curve and P-R curve.

Since we want to test to what extent our method preserves the semantic similarity, we just use the class labels as groundtruth, rather than the general Euclidean distances in most unsupervised hashing schemes.

We ran experiments to test our proposed method on two frequently-used image datasets: **MNIST** [33] and **CIFAR-10** [34].

A. MNIST

The **MNIST** [33] dataset has 70000 28*28 small greyscale images of handwritten digits from '0' to '9'. Each small image is represented by a 784-dimensional vector and a single digit label. Each 784-dimensional vector is stretched from a corresponding 28*28 greyscale image. We used 9000 random images as the training set, and 1000 random images as the query samples.

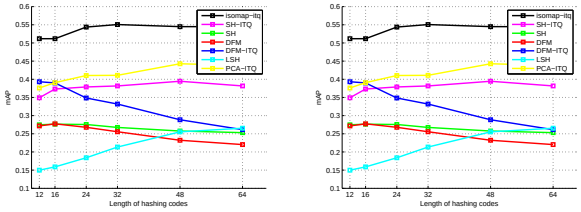

Fig. 1. MAP of MNIST(left) and CIFAR-10(right)

The MAP of each method with code lengths from 12 to 64 bits is reported in Fig. 1. We see that the proposed IsoMH performs best in all cases. We can infer from this experiment result that the global manifold learning method is much more comparative in this dataset. For other comparative methods, we see that ITQ can both improve the performance of hash code of SH and DH, but it improves SH much more than DH. With the code length increasing, the MAP of DFM and DFM-ITQ appear an decreasing trend while LSH and PCA-ITQ perform an increasing trend. And in shorter code, DH performs a little better than SH, while in longer codes, SH exceeds DH. However, as linear hashing scheme, PCA-ITQ performs better than SH and DH on this dataset, but is still inferior to IsoMH. And LSH is with the lowest MAP because it has no learning process on the training data.

The MAP figure is consistent with the P-R curves shown in Fig. 2. Fig. 2 also shows the precision and recall curves of hamming ranking. We see that linear methods especially PCA-ITQ achieve higher recall than manifold methods which have lower precision. And our IsoMH achieves higher recall than other methods while keeping competitive precision with SH and DH. Table I shows the training and indexing time of these test methods. Indexing time is the time of indexing all the 1000 points in the testing set. Because our IsoMH concludes shortest path algorithm, the time complexity is very high, and the training and indexing time is higher than others. From the training time of SH and SH-ITQ, or DH and DH-ITQ, we find the ITQ training doesn't take too much time. And the training and indexing time tend to increase as the code length increases.

B. CIFAR-10

CIFAR-10 [34] has 60000 32*32 subnail RGB images. These images are classified into 10 categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class has 6000 images. To retrieve more conveniently, every image in **CIFAR-10** is represented by a 320-dimensional GIST vector and a label from '1' to '10'. We also randomly pick 10000 images from the dataset and separate them into 9000-sample training set and 1000-sample test set.

The MAP on **CIFAR-10** dataset is represented in the right part of Fig. 1. We can see that PCA-ITQ has huge advantage over other methods in low bits code while our IsoMH exceeded it in higher bits. We can see that the MAP in **CIFAR-10** do not

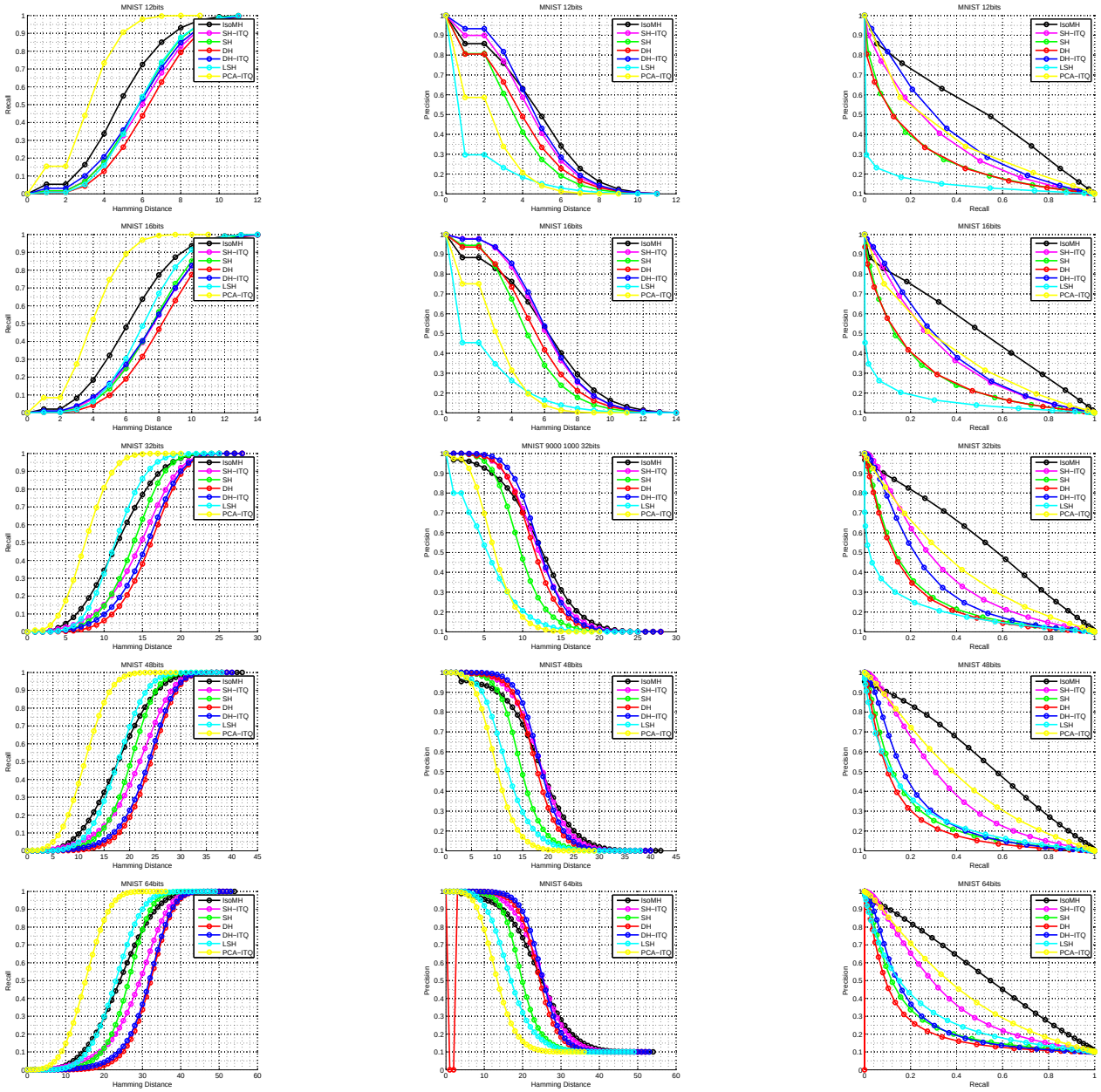

Fig. 2. Precision curves, recall curves and P-R curves on the MNIST dataset of all the methods in the experiment.

TABLE I
TIME CONSUMPTION (SECONDS) OF DIFFERENT METHODS ON THE MNIST DATASET. T MEANS TRAINING AND I MEANS INDEXING

Methods	12 bits		16 bits		32 bits		48 bits		64 bits	
	T	I	T	I	T	I	T	I	T	I
IsoMH	4197.8	22.1	4230.3	23.8	4248.9	23.2	4348.7	24.78	4385.7	26.5
SH-ITQ	0.29	0.14	0.30	0.14	0.50	0.25	0.77	0.41	1.09	0.59
SH	0.17	0.14	0.19	0.15	0.25	0.25	0.34	0.44	0.50	0.55
DH	4.38	0.15	4.55	0.17	4.19	0.25	4.75	0.42	4.82	0.54
DH-ITQ	4.41	0.16	4.73	0.14	4.43	0.25	4.97	0.40	5.20	0.51
LSH	0.01	0.15	0.01	0.15	0.02	0.26	0.01	0.43	0.02	0.50
PCA-ITQ	0.18	0.15	0.21	0.15	0.32	0.26	0.46	0.40	0.60	0.52

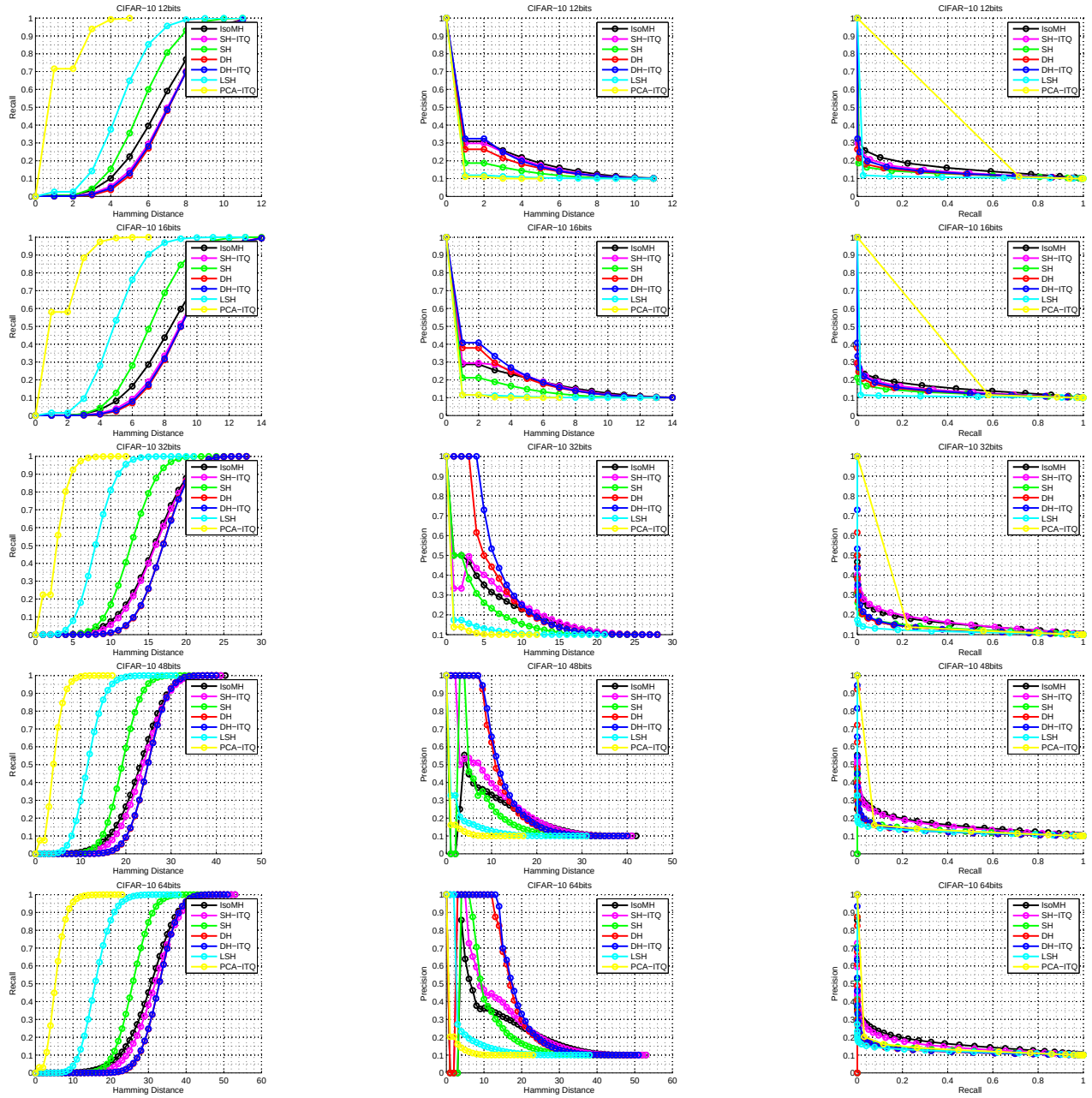

Fig. 3. Precision curves, recall curves and P-R curves on the CIFAR-10 dataset of all the methods in the experiment.

vary too much and is much lower than that **MNIST**, because the images of written digit are simple greyscale image without too much variation, so the data points of the same class in **MNIST** do not vary too much. But for images in **CIFAR-10**, the images are more complicated. The background of two images in the same class may vary a lot. What's more, a deer and a horse, or an automobile and a car may look the same, so it is hard to distinguish them.

The three kinds of curves are represented in Fig. 3. We can see that PCA-ITQ still is the top recall method, and LSH's recall is apparently better than other methods. The recall of SH is better than others manifold based hashing schemes, including SH-ITQ. The precision of DH and DH-ITQ is the highest among all the methods. Except for PCA-ITQ, the P-R curves of other methods are almost the same, and we can distinguish them better in MAP (Fig. 1).

Finally, the time consumption is represented in Table II. **CIFAR-10** and **MNIST** don't vary too much on training time. All our experiments were run on a 64-bit PC with 3.50 GHz Intel i7-4770K CPU and 16.0GB RAM.

V. Conclusion

We have proposed a hashing framework with global non-linear manifold learning method ISOMAP. By preserving the global geodesic distance into hash code, our IsoMH achieves

TABLE II
TIME CONSUMPTION (SECONDS) OF DIFFERENT METHODS ON THE CIFAR-10 DATASET. T MEANS TRAINING AND I MEANS INDEXING

Methods	12 bits		16 bits		32 bits		48 bits		64 bits	
	T	I	T	I	T	I	T	I	T	I
IsoMH	4189.57	11.66	4156.01	10.95	4083.98	11.59	4061.85	12.12	4079.17	12.68
SH-ITQ	0.19	0.14	0.18	0.13	0.34	0.24	0.54	0.37	0.79	0.49
SH	0.07	0.17	0.07	0.13	0.15	0.24	0.22	0.36	0.34	0.49
DH	2.64	0.16	2.27	0.12	2.29	0.24	2.28	0.36	2.33	0.46
DH-ITQ	2.70	0.12	2.36	0.12	2.51	0.24	2.59	0.35	2.74	0.49
LSH	0.02	0.14	0.01	0.14	0.01	0.24	0.01	0.35	0.01	0.47
PCA-ITQ	0.07	0.13	0.09	0.15	0.17	0.24	0.24	0.36	0.36	0.47

higher performance than other state-of-the-art linear or non-linear hashing schemes. However, IsoMH consumes more time in the training and indexing process than the other methods. We also try to add iterative quantization process in other traditional manifold learning based hashing schemes, and it improves the performance of those traditional methods to a large extent. Our work proves that non-linear manifold learning is significant in raising the retrieval performance. In the future, we will try to reduce the time complexity of our IsoMH while keeping competitive performance.

REFERENCES

[1] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," *Proceedings of the Neural Information Processing Systems Conference*, vol. 22, pp. 1509–1517, 2009.

[2] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *ICML*, 2011.

[3] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 2957–2964.

[4] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1092–1104, 2012.

[5] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *CVPR*, 2013.

[6] H. Yang, X. Bai, J. Zhou, P. Ren, Z. Zhang, and J. Cheng, "Adaptive object retrieval with kernel reconstructive hashing," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014, pp. 1955–1962.

[7] B. Xiao, Y. Haichuan, Z. Jun, R. Peng, and C. Jian, "Data-dependent hashing based on p-stable distribution." *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, vol. 23, no. 12, pp. 5033 – 5046, 2014.

[8] F. X. Yu, S. Kumar, Y. Gong, and S. F. Chang, "Circulant binary embedding," in *ICML*, 2014.

[9] X. Yan, H. Kaiming, K. Pushmeet, and S. Jian, "Sparse projections for high-dimensional binary codes," in *CVPR*, 2015.

[10] L. Cong, W. Jianxiang, C. Jian, B. Xiao, and L. Hanqing, "Online sketching hashing," in *CVPR*, 2015, pp. 2503–2511.

[11] X. Liu, Y. Mu, D. Zhang, B. Lang, and X. Li, "Large-scale unsupervised hashing with shared structure learning," *IEEE Transactions on Cybernetics*, vol. PP, no. 0, pp. 1–12, 2015.

[12] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proceedings of the Neural Information Processing Systems Conference*, 2009.

[13] M. Norouzi and D. M. Blei, "Minimal loss hashing for compact binary codes," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011.

[14] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3424–3431.

[15] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012.

[16] X. Liu, Y. Mu, B. Lang, and S.-F. Chang, "Mixed image-keyword query adaptive hashing over multilabel images," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 10, no. 2, pp. 22:1–22:21, Feb. 2014.

[17] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011.

[18] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, 1998, pp. 604–613.

[19] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *TPAMI*, vol. 35, no. 12, pp. 2916–2929, 2013.

[20] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[21] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 77–90, 2006.

[22] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering." in *NIPS*, vol. 14, 2001, pp. 585–591.

[23] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[24] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[25] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 21, pp. 7426–7431, 2005.

[26] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied & Computational Harmonic Analysis*, vol. 21, no. 1, p. 5C30, 2006.

[27] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS*, vol. 282, no. 3, 2008, pp. 1753–1760.

[28] T. Atsushi and A. Masaki, "Diffusion hashing," in *APSIPA ASC*, 2011.

[29] F. Shen, C. Shen, Q. Shi, A. Van Den Hengel, and Z. Tang, "Inductive hashing on manifolds," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 1562–1569.

[30] I. Go, L. Zhenguo, W. Xiao-Ming, and C. Shih-Fu, "Locally linear hashing for extracting non-linear manifolds," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014.

[31] T. F. Cox and M. A. Cox, *Multidimensional Scaling*. CRC Press, 2010.

[32] Y. Bengio, J.-f. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, "Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering," in *Advances in Neural Information Processing Systems*, 2004, pp. 177–184.

[33] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/

[34] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.