Graph PCA Hashing for Similarity Search

Xiaofeng Zhu, Xuelong Li, *Fellow, IEEE*, Shichao Zhang, Zongben Xu, Litao Yu, and Can Wang

***Abstract*—This paper proposes a new hashing framework to conduct similarity search via the following steps: first, employing linear clustering methods to obtain a set of representative data points and a set of landmarks of the big dataset; second, using the landmarks to generate a probability representation for each data point. The proposed probability representation method is further proved to preserve the neighborhood of each data point. Third, PCA is integrated with manifold learning to lean the hash functions using the probability representations of all representative data points. As a consequence, the proposed hashing method achieves efficient similarity search (with linear time complexity) and effective hashing performance and high generalization ability (simultaneously preserving two kinds of complementary similarity structures, i.e., local structures via manifold learning and global structures via PCA). Experimental results on four public datasets clearly demonstrate the advantages of our proposed method in terms of similarity search, compared to the state-of-the-art hashing methods.**

***Index Terms*—Hashing, image retrieval, manifold learning, similarity search, spectral clustering.**

I. Introduction

Similarity search is defined as quickly finding out the most similar data points from the big data set for a given query niques (such as kd-tree, B-tree, R-tree, M-tree, cover tree, metric tree, and QUC-tree) and hashing methods (such as data-dependent hashing and data-independent hashing) [6]–[10]. Index techniques conduct exact similarity search, but only effective for dealing with the low-dimensional data sets [11]–[13].

X. Zhu and S. Zhang are with the Guangxi Key Laboratory of MIMS and the College of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, China (e-mail: xfzhu0011@hotmail.com; zhangsc@mailbox.gxnu.edu.cn).

X. Li is with the Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China (e-mail: xuelong_li@opt.ac.cn).

Z. Xu is with the School of Mathematics and Statistics, Xian Jiaotong University, Xian 710049, China (e-mail: zbxu@mail.xjtu.edu.cn).

L. Yu is with the School of Information Technology and Electrical Engineering, University of Queensland, Queensland, QLD 4072, Australia (e-mail: l.yu4@uq.edu.au).

C. Wang is with the School of Information and Communication Technology, Griffith University, Southport, QLD 4215, Australia (e-mail: canwang613@ gmail.com).

By contrast, hashing techniques conduct Approximate Near-est Neighbor (ANN) search and have been becoming increas-ingly popular in real applications, such as multimedia search [14]–[16], database management [17], [18], and medical image analysis [19].

Different from exact similarity search, hashing needs less retrieval cost via scanning a subset of the entire data set to effectively conduct ANN search on high-dimensional data sets. The key step of hashing is hash function learning, which converts the high-dimensional continuous data into low-dimensional binary codes while preserving the similarity information (i.e., similarity structures of the data) among data points [20]–[22].

Previous methods of hash function learning include Locality Sensitive Hashing (LSH) based hashing methods [23]–[26], Principal Components Analysis (PCA) based hashing meth-ods [27]–[30], and manifold based hashing methods [30]–[35]. LSH based hashing methods randomly select a linear function as the hash function. The process of hash function learning is independent from the data, so LSH based hashing meth-ods are usually called data-independent hashing methods. PCA based hashing methods and manifold based hashing methods, respectively, utilize PCA and manifold techniques, to learn the hash functions. Moreover, PCA based methods and manifold based methods explore the data distribution to separately preserve the global and the local similarity structures of the data, and thus are called data-dependent hashing methods. In brief, LSH based hashing methods achieve fast retrieval speed, but need long bits to represent each data point for achieving reasonable hashing performance. Both PCA and manifold based hashing methods achieve significant hashing performance, but are with inefficient retrieval speed, i.e., at least quadratic time complexity for the training stage.

In this paper, we propose a linear hashing method to overcome the drawbacks of previous hashing methods. Specifically, we first employ linear spectral clustering methods to obtain the representative set and the landmark set of the big data set. We then use the landmark set to obtain the probability representation of the big data set, and also prove that the resulting probability presentation preserves the neighborhood of the data. We further combine PCA with manifold learning in a unified framework to learn hash functions using the representation set. As a result, the use of the representative set and the landmark set achieves a linear hashing, while both the

probability rep-resentation and the process of hash function learning enable to achieve high generalization ability and effective retrieval results.

Different from previous hashing methods, the proposed method has the following contributions:

1) The proposed method is a data-dependent method, which has been shown to outperform the data-independent hashing methods (such as LSH based hashing methods). Moreover, our method only needs representing the high-dimensional data by short binary codes (such as less than 64 bits in our experiments), while LSH based hashing methods need to represent each data point by long binary codes, such as more than 100-bit binary codes [21].
2) Even though the hashing methods (including PCA based methods, manifold based methods, and our proposed method) are data-dependent, our method achieves less computation cost, shorter binary codes and more effective hashing performance, than other data-dependent hashing methods. On one hand, the proposed method learns the hash functions using the landmark set, whose size is much small than the size of the big data set, so our hash function learning process needs less storage and computation costs. On the other hand, our proposed method learns the hash functions via preserving both the local and global similarity structures of the data, and thus enables to output significant hashing performance. By contrast, previous hashing methods only consider each of these two types of similarity structures, even though they have been shown to provide complimentary information to each other.
3) Experimental results in public data sets show that our proposed method achieves the best performance in terms of similarity search in large-scale data sets, compared to the state-of-the-art hashing methods. This verifies the theoretical advantages of our proposed method, i.e., concurrently preserving the global (via PCA) and the local (via manifold learning) similarity structures of the data.

II. Related Work

Hashing methods for ANN search usually include two steps, i.e., hash function learning and binarization. Hash function learning usually involves finding a continuous embedding of original data. Binarization converts the embedding of original data to binary codes by threshold methods. For example, the literatures in [36]–[38] simply quantize the low-dimensional embedded data into binary codes by a threshold, such as mean and median. Hierarchical hashing (HH) [39] first divides the real values of each dimension into four regions via three thresholds, and then encodes each dimension with double bits. Double-Bit Quantization (DBQ) [40] learns adaptive thresholds to quantize each projected dimension into double bits. However, above methods separately learn the hashing functions and the threshold, so that it is difficult for such sequential two-step methods to receive the optimal hashing performance even though each of two steps achieves their individual optimizations [33]. To address this issue, the method in [33] simultaneously learns the hash functions and the threshold to output the optimal results of both hash functions and the threshold.

In the literature, most hashing methods focus on the step of learning hash function. Therefore, in this section, we categorize previous hashing methods into three categories, such as uni-modal hashing, multi-modal hashing, and cross-modal hashing, according to the number of the data sets to learn hash functions [8], [41].

Uni-modal hashing conducts ANN search on a data set and can be categorized into three sub-categories, such as unsupervised hashing, supervised hashing and semi-supervised hashing. Unsupervised hashing learns the hash functions without taking prior knowledge (i.e., label information) into account [32], [42]–[44]. Supervised hashing learns the hash functions via considering prior knowledge, such as "similar" and "dissimilar" pairs of the data [36], [45], [46]. Semi-supervised hashing employs a supervised term to minimize the empirical error on the labeled data and an unsupervised term to maximize desirable properties, such as variance and independence of individual bit in the binary codes [30], [34].

This paper focuses on unsupervised hashing since it is expensive to obtain prior knowledge for the construction of hashing in real applications. We further partition current unsupervised hashing methods into two sub-groups, such as data-independent hashing methods (such as LSH based hashing) and data-dependent hashing methods (such as PCA based hashing methods and manifold based hashing methods). Data-independent hashing methods generate hash functions without considering the properties of the data. For example, LSH and its variants [17], [23], [24], [47] learn the hash functions based on random projections. The Kernelized LSH (KLSH) [25] captures the intrinsic relationships among training samples using kernel functions instead of linear inner products. Shift-invariant Kernel LSH (SKLSH) [26] learns the hash functions by mapping the random features into shift-invariant kernels. Data-dependent hashing methods have been becoming increasingly popular in the applications of ANN search since it can generate effective hash functions by exploring the properties of training data. For example, PCA based hashing methods [28], [48] learn the hash functions via preserving the maximal covariance of original data and have been shown to outperform LSH based hashing methods in [29]. However, PCA based hashing methods assign the same weight to each bit, even though different bits contain different variance. To address this issue, Isotropic hashing (IsoHash) learns the hash functions for producing projected dimensions with isotropic variances (equal variances) [39], while ITerative Quantization (ITQ) [29] designs a non-orthogonal relaxation or sequential projection to alleviate such an unbalanced variance issue. Manifold based hashing methods [31], [35], [44], [49] employ manifold learning techniques to learn hash functions, for preserving the local similarity structures of the data, i.e., similar data points have similar binary codes. The key issues of most manifold based hashing methods are the high time complexity [32] and the out-of-sample problem (i.e., not generating explicit hash functions), such as the methods [32], [44], [50].

Multi-modal hashing is designed to conduct hash function learning for encoding multi-modal data [16], [51]. For example,

the method [16] first uses an iterative method to preserve the semantic similarities among training examples, and then keeps the consistency between the hash codes and the corresponding hashing functions designed for multiple modals. Multiple Feature Hashing (MFH) [20] preserves the local structure information of each modal and also globally considers the alignments for all the modals to learn a group of hash functions for real-time large scale near-duplicate web video retrieval. However, multi-modal hashing is constraint to the real applications because it is difficult and impractical to obtain multi-modal queries.

Cross-modal hashing is much more popular because only one view is needed for a query. For example, in real applications, we can use search engines (such as Google, Bing, and Baidu) to conduct cross-modal search, such as searching images via a text query. Besides, the retrieval tasks (such as image-to-image, text-to-image, and image-to-text) can also be conducted by cross-modal hashing. In the literatures, Multi-modal Latent Binary Embedding (MLBE) [52] devises a generative model to conduct hash function learning using only hundreds of data points out of millions data points as training samples, i.e., a small subset of training data. Both Cross-Modal Similarity Sensitive Hashing (CMSSH) [16] and Inter-Media Hashing (IMH) [2] are not used to learn the hash functions from large-scale training data sets yet since they need high time complexity, such as $O(n^3)$ where *n* is the sample size. To reduce the time complexity, Linear Cross-Modal Hashing (LCMH) [13] takes both the intra-similarity in each modal and the inter-similarity across the modals to conduct cross-modal hashing within linear time complexity. Recently, Latent Semantic Sparse Hashing (LSSH) [53] conducts cross-modal similarity search by employing sparse coding and matrix factorization, while Deep Cross-Modal Hashing (DCMH) [54] integrates feature learning with hash function learning in a framework.

III. Preliminary

In this section, we first show the notions used in this paper, and then separately introduce the background of PCA based hashing and manifold based hashing, followed by the motivation of our proposed method.

A. Notations

In this paper, we denote matrices, vectors, and scalars, respectively, as boldface uppercase letters, boldface lowercase letters, and normal italic letters. We summarize other notations used in this paper in Table I.

Given a set of n data points $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n\} \in \mathbb{R}^{n \times d}$ where *d* is the number of features, we assume that the matrix $\mathbf{Z}$ is zero-centered, i.e., $\sum_{j=1}^{n} \mathbf{z}_{j,i} = 0$ $(i = 1, ..., d)$. The goal of this paper is to learn a binary code matrix $\mathbf{B} \in \{-1, 1\}^{n \times c}$ for $\mathbf{Z}$ without label information to conduct uni-modal unsupervised hashing, where c denotes the number of the codes. Specifically, for the *k*-th bit $(k = 1, ..., c)$, we denote its binary encoding function as $h_k(\mathbf{z}) = sgn(\mathbf{z}\mathbf{w}_k)$, and

$$\mathrm{sgn}(v) = \begin{cases} 1, & \text{if } v \geq 0, \\ -1, & \text{otherwise.} \end{cases} \tag{1}$$

TABLE I
Details of Used Notations in This Paper

$\mathbf{Z}$	The feature matrix of a data set
$\mathbf{X}$	The feature matrix of the representative set
$\mathbf{x}$	a vector of $\mathbf{X}$
$\mathbf{x}^i$	the *i*-th row of $\mathbf{X}$
$\mathbf{x}_j$	the *j*-th column of $\mathbf{X}$
$x_{i,j}$	the element in the *i*-th row and the *j*-th column of $\mathbf{X}$
$\|\|\mathbf{X}\|\|_F$	the Frobenius norm of $\mathbf{X}$, i.e., $\|\|\mathbf{X}\|\|_F = \sqrt{\sum_{i,j} x_{i,j}^2}$
$\|\|\mathbf{X}\|\|_{2,1}$	the $\ell_{2,1}$-norm of $\mathbf{X}$, i.e., $\|\|\mathbf{X}\|\|_{2,1} = \sum_i \sqrt{\sum_j x_{i,j}^2}$
$rank(\mathbf{X})$	the rank of $\mathbf{X}$
$\mathbf{X}^T$	the transpose of $\mathbf{X}$
$\mathrm{tr}(\mathbf{X})$	the trace of $\mathbf{X}$
$\mathbf{X}^{-1}$	the inverse of $\mathbf{X}$

where $sgn(\cdot)$ denotes the result of element-wise application and $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_d] \in \mathbb{R}^{d \times c}$ is the transformation matrix. Therefore, we have the following formulation:

$$\mathbf{B} = \mathrm{sgn}(\mathbf{Z}\mathbf{W}). \tag{2}$$

B. PCA Hashing

According to the point of view of the information-theory, a hashing should maximize the information provided by each bit [30], i.e., a binary bit giving balanced partitioning of $\mathbf{Z}$ may produce maximum information. However, it is difficult to find such hash functions to meet the balancing requirement. To do this, PCA based hashing first assumes that the maximum entropy partitioning implies to maximize the variance of each bit. That is, the variance of each bit is maximized and the bits are pairwise uncorrelated as well [29], [30]. As a consequence, this assumption results in the following formulation:

$$\begin{aligned} \Gamma(W) &= \sum_k \mathrm{var}(h_k(\mathbf{z})) \\ &= \sum_k \mathrm{var}(\mathrm{sgn}(\mathbf{z}\mathbf{w}_k)) \\ s.t., \tfrac{1}{n}\mathbf{B}^T\mathbf{B} &= \mathbf{I}_c \end{aligned} \tag{3}$$

where $\mathrm{var}(\cdot)$ is the variance of a vector (or a matrix), and $\mathbf{I}_c \in \mathbf{R}^{c \times c}$ is an identity matrix.

The maximized variance in (3) indicates that the encoding functions produce exactly balanced bits, i.e., $\mathbf{B}^T\mathbf{B} = \mathbf{I}_c$. However, the discrete issue makes (3) intractable. To address this issue, (3) is usually transferred to the following continuous version:

$$\begin{aligned} \tilde{\Gamma}(\mathbf{W}) &= \tfrac{1}{n}\sum_k \mathbf{w}_k^T\mathbf{Z}^T\mathbf{Z}\mathbf{w}_k \\ &= \tfrac{1}{n}\mathrm{tr}(\mathbf{W}^T\mathbf{Z}^T\mathbf{Z}\mathbf{W}), \\ \text{s.t.,}\ \mathbf{W}^T\mathbf{W} &= \mathbf{I}_c \end{aligned} \tag{4}$$

where the orthogonal constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}_c$ requires that the vectors of $\mathbf{W}$ (the hyperplanes of the hash functions) are uncorrelated (or orthogonal to each other).

Equation (4) is exactly the PCA, so we call the resulting hashing based on (4) and its variants as PCA based hashing methods, and can obtain the optimal $\mathbf{W}$ as the top *c* eigenvectors of the data covariance matrix $\mathbf{Z}^T\mathbf{Z}$, which corresponds to the

top c non-zero eigenvalues of $\mathbf{Z}^T\mathbf{Z}$. The PCA hashing in (4) has been demonstrated to preserve the global similarity structures of the data [33]. It is noteworthy that similarity preserving is the key assumption of hash function learning, i.e., converting original data to the low-dimensional binary codes [55].

The PCA based hashing methods need high time complexity of the training stage, i.e., quadratic to the sample size, to conduct hash function learning. Moreover, the variance of the data in different PCA directions is usually different. In particular, the directions with high variance carry more information than the directions with low variance. To deal with this issue, Spectral Hashing (SH) [27] employs a separable Laplacian eigenfunction to forbid assigning more bits to the directions with less variance. However, SH is a heuristic method by assuming the data to follow uniform distribution. Different from SH, Semi-Supervised Hashing (SSH) [30] sequentially projects the directions via relaxing the orthogonality constraints of PCA. ITerative Quantization (ITQ) searches a rotation of zero-centered data to minimize the quantization error to keep the variance of the data.

C. Manifold Hashing

Different from the PCA based hashing methods [27]–[30] to preserve the $global$ similarity structures of all the data, manifold based hashing methods [21][33] focus on the *local* similarity structures, i.e., k Nearest Neighborhood (kNN) preservation for each data point. Their motivation is that 1) the requirement of some real applications. For example, conducting information retrieval via search engines such as Google and Bing, usually expects finding the most similar results (i.e., a subset of the big data set) for a given query; 2) the requirement of manifold learning. A lot of literatures have shown that the global similarity structure preservation possibly results in bad performance due to the adverse impact of noise and outliers, which will be avoided by the local similarity preservation [56].

By following the literatures such as [56] to preserve the local similarity for each data point, we first build a similarity matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ using the following steps:

1) *Constructing the adjacency graph:* We calculate the similarity (or the distance) between any two data points.
2) *Constructing the sparse adjacency graph:* We keep the similarity of two data points if a data point is one of the k nearest neighbors (where k is a tuning parameter) of another data point, otherwise 0. This enables to keep noise or outliers out of the local similarity preservation.

Above steps output a sparse similarity matrix for the data set. After this, we also expect to minimize the weighted average Hamming distance of each data pair, and thus devise the following objective function:

$$\sum_{i,j=1}^{n} s_{i,j}\|\mathbf{b}_i - \mathbf{b}_j\|^2, \text{s.t.}, \mathbf{B}^T\mathbf{B} = \mathbf{I}_c, \mathbf{B}^T\mathbf{1} = \mathbf{0} \qquad (5)$$

where each row of $\mathbf{S}$ indicates that each data point is represented by a subset of all data points. Different from the global similarity structures in (4), the local similarity structures in (5) are preserved by a sparse kNN graph, which results in a significant reduction of the storage and computation costs.

We define a diagonal matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$ whose entries are defined as $\mathbf{D}_{i,i} = \sum_{j=1}^{n} s_{ij}$, and let $\mathbf{L} = \mathbf{D} - \mathbf{S}$, (5) is thus converted to the following objective function:

$$\min_{\mathbf{B}} \text{Tr}(\mathbf{B}^T\mathbf{L}\mathbf{B}) \text{ s.t.}, \mathbf{B}^T\mathbf{B} = \mathbf{I}_c, \mathbf{B}^T\mathbf{1} = \mathbf{0} \qquad (6)$$

where $\mathbf{1}$ and $\mathbf{0}$, respectively, are the matrices with all entries as 1 and 0.

The manifold based hashing methods (such as Self-Taught Hashing (STH) [57] and Anchor Graph Hashing (AGH) [31]) were designed to automatically preserve the local neighborhood structures inherent in the data to learn compact binary codes. Hence, manifold based hashing methods have been shown to achieve more effective hashing performance than LSH based hashing methods and PCA based hashing methods, but need more expensive time cost than these two kinds of hashing methods.

D. Motivation

By comparing two eigenvalue problems in (4) and (6), they separately preserve the global similarity structures and the local similarity structures of the data, to result in the time complexity at least $\min\{O(n^2d, d^3)\}$ and $O(n^3)$, respectively.

On one hand, either PCA based hashing methods or manifold based hashing methods only preserves one type of similarity structures. However, in the literatures, both of these two kinds of similarity structures have been demonstrated to strengthen the performance of unsupervised spectral feature selection due to providing complimentary information to each other [55], [58]. Specifically, the global similarity structure preservation contains generalization ability, while the local similarity structure preserves the manifolds of the data. It is noteworthy that the process of hash function learning is actually a process of dimensionality reduction. Thus, it is reasonable to combine these two types of similarity structures into a framework.

On the other hand, if the feature dimensions d is large, the high time complexity of these two hashing techniques are forbidden in the applications of big data. Moreover, big data are usually stored in the disk rather than the memory of the PC. The retrieval in the disk is time-consuming. This motivates us to select a subset of big data (called the representative set in this paper), which can be put into the memory of the PC, to learn the hash functions.

Therefore, in this paper, we integrate PCA with manifold learning to simultaneously preserve the local and global similarity structures of the data to learn the hash functions on the representative set with linear time complexity. First, we conduct linear spectral clustering to hierarchically conduct clustering on the big data set because the big data may not be fed into the memory of modern PC for similarity search. Specifically, we employ the spectral clustering method in [59] to output a set of cluster centers, which can be fed into the memory of PC and approximately represent the distribution of the big data set. We regard the resulting cluster centers as the representative set of the big data set. We then employ the same linear clustering method on the representative set to further output their cluster centers, which are called the landmarks of the representative

set (and also the big data set) in this paper. Second, we use the landmark set (containing all the landmarks) to generate a probability representation for each data point of the big data set, and further prove that the probability representations preserve the neighborhood of the big data set. Third, we use the probability representations of the representative set to learn the hash functions by integrating the local similarity structure preservation (via PCA) with the global similarity structure preservation (via manifold learning) in a framework. As a consequence, the training stage outputs the landmark set, the hashing functions, and the binary codes of the big data set.

In testing stage, we first yield the probability representation of each query via the landmark set, and then transfer the resulting probability representation to its binary codes via the hash functions. Finally, we conduct similarity search between binary codes of the query and the big data set via calculating their Hamming distance in the memory of PC.

IV. Approach

A. Graph PCA

We first transfer (4) via ignoring the factor $\frac{1}{n}$ to yield the following objective function:

$$\max \mathrm{tr}(\mathbf{W}^T\mathbf{Z}^T\mathbf{Z}\mathbf{W}),\ \mathrm{s.t.},\mathbf{W}^T\mathbf{W}=\mathbf{I}_c. \tag{7}$$

We then assume that the binary matrix $\mathbf{B}$ can be transferred as a combination between the feature matrix $\mathbf{X}$ and a linear transformation matrix $\mathbf{W}$, i.e., $\mathbf{B}=\mathbf{X}\mathbf{W}$. We further transfer (6) to obtain the following formulation:

$$\min_{\mathbf{W}} \mathrm{tr}(\mathbf{W}^T\mathbf{Z}^T\mathbf{L}\mathbf{Z}\mathbf{W}),\ \mathrm{s.t.},\mathbf{W}^T\mathbf{W}=\mathbf{I}_c. \tag{8}$$

By integrating (7) with (8), we have the objective function as follows:

$$\begin{aligned}&\min \mathrm{tr}(\mathbf{W}^T\mathbf{Z}^T\mathbf{L}\mathbf{Z}\mathbf{W})-\alpha\mathrm{tr}(\mathbf{W}^T\mathbf{Z}^T\mathbf{Z}\mathbf{W}),\\ &\mathrm{s.t.},\mathbf{W}^T\mathbf{W}=\mathbf{I}_c\end{aligned} \tag{9}$$

where α is a tuning parameter. Finally, we have our final objective function to learn the hash functions as follows:

$$\begin{aligned}&\min \mathrm{tr}(\mathbf{W}^T(\mathbf{Z}^T(\mathbf{L}-\alpha\mathbf{I}_n)\mathbf{Z})\mathbf{W}),\\ &\mathrm{s.t.},\mathbf{W}^T\mathbf{W}=\mathbf{I}_c\end{aligned} \tag{10}$$

where $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is an identity matrix. (10) integrates the minimum of the local similarity structure preservation in (9) with the maximum of the global similarity structure preservation in (8) to achieve two kinds of similarity structure preservation at the same time, where the tuning parameter α is used to balance their weights.

Although (10) can simultaneously learn the local and global structures of the data, its time complexity is still high, i.e., $\min\{O(n^2d), O(d^3)\}$, whose complexity is same as that of PCA hashing in (4). In the next subsection, we design to learn the representative set and the landmark set to represent the distribution of the big data set.

B. Probability Representation

Manifold based hashing methods [20], [34], [57], [60] generate semantic representations of original data, instead of using original *d* feature spaces, to represent each data point for preserving the local similarity structures of the data. More specifically, manifold based hashing methods assume that each data point form a feature space, so that there are *n* feature spaces. In this way, each data point represented by all the data points obtains a probability or a coordinate in each feature space. Moreover, each data point is sparsely represented by a subset of all data points to avoid the adverse impact of noise or outliers. As a result, the resulting representation naturally preserves the local similarity structures among data points.

However, such a method has the following limitations to be addressed. First, the time complexity to construct the representation is at least $O(n^3)$, which is overwhelming for big data. Second, given the resulting similarity matrix $\mathbf{S}$ in (5), we assume that all the data points lie in an *n* dimensional feature space. Actually, in big data, the value of *n* is very large, and thus resulting in a very high-dimensional data matrix, which needs expensive storage and computation costs. Third, the high-dimensional data have been demonstrated to have a low-dimensional feature space where noise and redundancy are removed [55]. Finally, the sample size of the data set is usually too big to fit them in the memory of PC. In this case, it will be very time consuming to learn the hash functions.

In this paper, we propose a novel strategy to address the above issues, by which an efficient scheme of hash function learning is developed without using the whole data set for local similarity structure preservation. First, we use linear clustering methods to group the data set into subgroups whose cluster centers consist of the representative set of the data set. The number of the cluster centers can be large so that the representative set can be fed into the memory of the PC. Second, we usually generate a representative set as large as possible to capture the characteristics of the data set, but it is still time consuming to conduct hash function learning with the similarity matrix constructed by the representative set. Hence, we further conduct clustering on the representative set to obtain the cluster centers, i.e., the landmarks, each of which is the cluster center of the representative set. Moreover, the number of cluster centers is regarded as the dimensions of the low-dimensional feature space of original data. Third, given the dimensions of the intrinsic space, we use the landmark set to probabilistically represent each data point of the big data set and the representative set as well. Finally, we prove that the resulting representation preserves the local similarity structures.

We have at least two methods to search the dimensions of the intrinsic space via conducting clustering on the high-dimensional data, such as the method of affinity propagation clustering [61] and the cross-validation method. However, the algorithmic complexity of affinity propagation is quadratic in the number of data points and thus cannot be used to deal with big data. In this paper, we predefine a set of dimensions and then use the cross-validation methods to find the best number of clusters for the data sets. Specifically, given the dimensions of the intrinsic space of the original high-dimensional similarity matrix, i.e., *m*, the further step is to partition the feature space of the similarity matrix into clusters, which is actually a group partition issue.

In this paper, we first employ the clustering method in [59] with linear complexity to generate *r* cluster centers, which consist of the representative set $\mathbf{X} \in \mathbb{R}^{r\times d}$. Given the representative set $\mathbf{X}$, we then employ the same clustering method to generate *m* clusters and regard the resulting cluster centers as the landmarks, i.e., $\mathbf{l}_k$ ($k = 1, 2, ..., m$). We further devise a probability representation method to represent each data point by the landmarks. Specifically, for a given representative set $\mathbf{X}$, each data point $\mathbf{x}_i$ lies in the *m*-dimensional space spanned by the landmark set. Moreover, each data point has a probability belonging to each space. Given a query $\mathbf{y}$, we want to find the data points which have similar locations to $\mathbf{y}$ in the space of $\mathbf{X}$. That is, we also give the probability of $\mathbf{y}$ to each space spanned by the landmarks. In this way, finding the similar data points is transferred to find the minimal distance of probability between the query and each data point in the data set. Actually, we cannot obtain exact probability of the data set without infinity data points, but we can approximately estimate it via the following steps.

First, we calculate the Euclidean distance between each data point $\mathbf{x}_i$ and each cluster center $\mathbf{l}_k$, i.e.,

$$u_{i,k} = \|\mathbf{x}_i - \mathbf{l}_k\|_2^2. \tag{11}$$

According to the literature [62], the probability of a data point belonging to a space can be defined as the Euclidean distance $g_{i,k}$

$$g_{i,k} = \frac{\exp(-u_{i,k}/\sigma)}{\sum_{j=1}^{m}\exp(-u_{i,j}/\sigma)} \tag{12}$$

where σ is a tuning parameter to control the decay rate of $g_{i,k}$ with respect to $u_{i,k}$.

Let $\mathbf{g}_i = [g_{i,1}; ...; g_{i,k}; ...; g_{i,m}]$, $\mathbf{g}_i$ forms a probability representation of $\mathbf{x}_i$, which characterizes the probability distribution of the location of $\mathbf{x}_i$ in the space spanned by $\mathbf{X}$. Actually, the rationale of the probability definition of $\mathbf{g}_i$ is similar to that of kernel density estimation with a heat kernel. That is, if $\mathbf{x}_i$ is near the k-th cluster center, $g_{i,k}$ will have large value. Otherwise, the value of $g_{i,k}$ will small. Moreover, the reduce rate will be decided by the tune parameter σ, which is set as $\sigma = 1$ in this paper.

Second, in the traditional similarity matrix $\mathbf{S}$, each data point is represented by a subset of all data points to avoid the effect of noise. Hence, we also use a subset of landmarks (i.e., s) to represent each data point. That is, in the local similarity preservation, we only preserve first s largest probabilities in $\mathbf{g}_i$, and set the left probabilities to 0. Specifically, we denote g_s as the s-th most largest probability of $\mathbf{g}_i$ to have

$$\hat{g}_{i,k} = \begin{cases} g_{i,k} & \text{if } g_{i,k} \geq g_s \\ 0 & \text{if } g_{i,k} < g_s. \end{cases} \tag{13}$$

After this, we normalize $\hat{g}_{i,k}$ to have the following equation:

$$\tilde{g}_{i,k} = \frac{\hat{g}_{i,k}}{\sum_{j=1}^{m}\hat{g}_{i,k}}. \tag{14}$$

Finally, we set the probability representation of $\mathbf{x}_i$ as $\tilde{\mathbf{g}}_i = [\tilde{g}_{i,1}; ...; \tilde{g}_{i,k}; ...; \tilde{g}_{i,m}]$.

The probability representation $\tilde{\mathbf{g}}_i$ is a sparse vector, which characterizes the spatial structure of $\mathbf{x}_i$ in the space spanned by the representative set $\mathbf{X}$ or the big data set $\mathbf{Z}$. In this case, similar data points have similar probability representations, so $\tilde{\mathbf{g}}_i$ is a reasonable representation of $\mathbf{x}_i$. In the left of this section, we use Theorem 1 to show that the proposed m-dimensional representation model preserves the local similarity structures of training data.

Theorem 1: Following the process from (11) to (14), the neighborhood of each data point in either the representative set $\mathbf{X}$ or the data set $\mathbf{Z}$ is preserved.

Proof. Here we focus on the proof of the neighborhood preservation of the representative set, and the neighborhood preservation of the big data set can be proved with the similar principle. We first add these m cluster centers into $\mathbf{X}$ to form a new dataset $\mathbf{X}^{'}$, i.e.,

$$\begin{aligned}\mathbf{X}^{'} &= \{\mathbf{x}_1^{'}, \mathbf{x}_2^{'}, \ldots, \mathbf{x}_r^{'}, \mathbf{x}_{r+1}^{'}, \ldots, \mathbf{x}_{r+m}^{'}\} \\ &= \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_r, \mathbf{l}_1, \ldots, \mathbf{l}_m\}.\end{aligned} \tag{15}$$

We then denote each data point in $\mathbf{X}$ as

$$\mathbf{x}^{'} = \sum_{j=1}^{r+m}\tilde{g}_{i,j}\mathbf{x}_j^{'} \tag{16}$$

where $j = 1, \ldots, r + m$, and $\tilde{\mathbf{g}}_{i,j} = 0$ if $\mathbf{x}_j^{'}$ is not one of the *m*-nearest cluster centers of $\mathbf{x}_i^{'}$.

According to the literature [63], each data point in $\mathbf{X}^{'}$ can be represented by m cluster centers as in (16). Obviously, this is also applied to $\mathbf{X}$ since $\mathbf{X}$ is a subset of $\mathbf{X}^{'}$. Actually, (16) leads to the following reconstruction (or representation) error measured by the following cost function:

$$\mathbf{E}(\mathbf{X}) = \textstyle\sum_i \left|\mathbf{x} - \sum_j \tilde{g}_{i,j}\mathbf{x}_j^{'}\right|. \tag{17}$$

Moreover, (16) shows two characteristics, i.e., sparseness and invariance. The sparseness means that each representative data point $\mathbf{x}_i$ is represented by a subset of m cluster centers (i.e., s nearest neighbors) via enforcing $\tilde{g}_{i,j} = 0$ if $\mathbf{x}_j^{'}$ does not belong to this set. According to the literature [63], the defined $\tilde{g}_{i,j}$ in (14) minimizes the reconstruction errors in (16) as well as obeys several important symmetries. That is, the new representation of $\mathbf{x}_i^{'}$ in $\mathbf{X}_i^{'}$ is invariant to rotations, rescalings, and translations. More specifically, (15) shows the invariance to rotations and rescalings, and the constraint $\sum_j \tilde{g}_{i,j} = 1$ ensures the invariance to translations. Moreover, these symmetries ensure that the reconstruction weights (i.e., the new representations) make geometric properties independent on a particular distribution of the data. Therefore, the similarity structures of the data points in $\mathbf{X}$ can be well preserved in the derived *m*-dimensional representation. ■

C. Hash Function Learning

In the proposed hashing method, namely graph PCA (gPCA), we first conduct Section IV-B to yield the probability representation of each data point via Algorithm 1, and then optimize the objective function in (10) to obtain $\mathbf{W}$. Finally, the binary codes of the data points can be obtained by $sgn(\tilde{\mathbf{G}}\mathbf{W})$. We list the

Algorithm 1: The Pseudo of Calculating the Probability Representation of All Data Points

Input: $\mathbf{y} \in \mathbb{R}^d$; $\mathbf{Z} \in \mathbb{R}^{n \times d}$; α: tuning parameter; s: the Value of kNN; m: the number of clusters.
Output: $\tilde{\mathbf{g}}_y \in \mathbb{R}^m$ and $\tilde{\mathbf{G}} \in \mathbb{R}^{n \times m}$.

1: Conduct clustering to generate $\mathbf{X}$ via [59];
2: Conduct clustering to generate the landmarks via [59];
3: Calculate Euclidean distance of any data pair in $\mathbf{Z}$ via (11);
4: Calculate the probability of each data point in $\mathbf{Z}$ via (12);
5: Calculate the sparse probability representation of each data point in $\mathbf{Z}$ via (13);
6: Normalize the sparse probability representation of each data point in $\mathbf{Z}$ via (14);

Algorithm 2: Training Stage of the Proposed Method gPCA

Input: $\mathbf{Z} \in \mathbb{R}^{n \times d}$; α: tuning parameter; c: the number of bits.
Output: $\mathbf{B} \in \mathbb{R}^{n \times c}$; $\mathbf{W} \in \mathbb{R}^{m \times c}$.

1: Yield the probability representation $\hat{\mathbf{G}}$ of $\mathbf{X}$ via Algorithm 1;
2: Conduct eigenvalue decomposition on $(\hat{\mathbf{G}}^T(\mathbf{L} - \alpha\mathbf{I}_n)\hat{\mathbf{G}})$;
3: Yield the probability representation $\tilde{\mathbf{G}}$ of $\mathbf{X}$ via Algorithm 1;
4: Calculate $\mathbf{B}$ via $sgn(\tilde{\mathbf{G}}\mathbf{W})$ for all the data points of $\mathbf{Z}$;

Algorithm 3: Testing Stage of the Proposed Method gPCA

Input: $\mathbf{y} \in \mathbb{R}^d$; $\mathbf{W} \in \mathbb{R}^{m \times c}$.
Output: $\mathbf{b}_y \in \mathbb{R}^c$.

1: Yield the probability representation $\tilde{\mathbf{g}}_g$ of $\mathbf{y}$ via Algorithm 1;
2: Calculate $\mathbf{b}_y$ via $sgn(\tilde{\mathbf{g}}_y\mathbf{W})$;
3: Calculate the Hamming distance between $\mathbf{b}_y$ and $\mathbf{B}$;

pseudo of training stage and testing stage, respectively, of our proposed method, in Algorithm 2 and Algorithm 3.

The time cost of our method focuses on the clustering and the eigenvalue decomposition. The clustering method is linear (i.e., $O(dmn)$) and the eigenvalue decomposition is quadratic to the number of landmarks, i.e., $\min\{O(m^2d), O(d^3)\}$. Hence, the time complexity of our method is $O(dmn)$ ($d \ll n$). We list the time complexity of training stage and testing stage of some popular hashing methods in Table II.

The space complexity of our gPCA method is $O(d(m+n))$ in training stage and $O(dl + ck)$ plus $O(cn)$ (binary bits) in testing stage.

D. Connection to Previous Hashing Methods

Comparing with data-independent hashing methods, i.e., LSH based hashing methods in [17], [23]–[26], [47], our proposed gPCA learns the hash functions according to the data distribution, so that outputting reasonable hashing performance. Moreover, our gPCA can use shorter binary codes (i.e., less than 64 bits) to represent each data point, than LSH based hashing. Although our gPCA has higher time complexity than LSH based hashing methods, in terms of training stage and testing stage, our gPCA achieves linear time complexity for training stage and constant time complexity for testing stage, and thus can be easily run in the modern PC for conducting similarity research on big data.

Comparing with PCA based hashing methods in [28]–[30], [48] which only consider to preserve the maximum variance of the data (i.e., the global similarity structures), our proposed gPCA also considers the local similarity preservation. Moreover, these current PCA based hashing methods have higher time complexity than our methods. Comparing with manifold based hashing methods in [2], [20], [57], [60] which only consider to preserve the local similarity structures, our proposed gPCA also considers the global similarity preservation by preserving the maximum variance of the data. Moreover, these previous manifold based hashing methods need at least quadratic time complexity for training stage, while our method only needs linear time complexity. It is noteworthy that both PCA based hashing methods and manifold based hashing methods are a special issue of our proposed gPCA methods. For example, our method shrinks to a manifold based hashing methods by setting $\alpha = 0$ in (10), while our method becomes a PCA based hashing method by setting the value of α as infinity.

Actually, the literatures [58], [64] have shown that both the local and global similarity structures of the data provide complimentary information to each other so that achieving optimal performance. This indicates that it is reasonable for our method to combine these two kinds of geometries together. Locally Linear Hashing (LLH) [35] is the only hashing method which considers both of these geometries to conduct hash function learning. However, LLH usually achieves suboptimal hashing results because it sequentially conducts two steps to preserve these two similarity structures. By contrast, our proposed gPCA method considers preserving them simultaneously and thus yielding optimal hashing results. Furthermore, our method only needs to conduct eigenvalue decomposition once while LLH needs to do that twice.

TABLE II
TIME COMPLEXITY COMPARISON OF THE HASHING METHODS

	Training stage	Testing stage
LSH [65]	$O(1)$	$O(1)$
SH [27]	$O(d^2n)$	$O(dc)$
ITQ [29]	$O(d^2n + c^3)$	$O(dc)$
AGH [31]	$O(dmn + m^2n)$	$O(dm)$
MDSH [28]	$O(d^2n)$	$O(dc)$
LLH [35]	$O(dn)$	$O(sc)$
IMH [32]	$O(dmn)$	$O(dm)$
gPCA	$O(dmn)$	$O(dm)$

Note that c, d, m, n, and s, respectively, are the length of binary codes, the dimensions of the data, the number of centers, the training size, and the number of nearest neighbors.

V. Experimental Results

In this section, we evaluate the proposed gPCA method on four widely used benchmark data sets, i.e., CIFAR (60K) [66], MNIST (70K) [59], NUS-WIDE (193K) [67], and GIST (500K) [14]. We compare our gPCA with the data-independent hashing method LSH [47], two PCA based hashing methods (i.e., MDSH [28] and ITQ [29]), two manifold based hashing methods (such as AGH [31] and IMH [32]), and the method LLH [35] which sequentially preserves the local and the global similarity structures of the data.

A. Comparison Methods

We list the details of the comparison methods as follows:

1) LSH randomly selects linear functions as hash functions. In our experiments, we follow the literature [33] to generate a Gaussian random matrix as the hash functions.
2) MDSH first learns the similarity matrix of original data to preserve the global similarity structures of the data, and then optimizes a matrix factorization problem on the resulting similarity matrix to output the top eigenvectors as the hash functions.
3) ITQ first employs PCA to preserve the global similarity structures of original data, and then learns an orthogonal transformation matrix as the hash functions via solving the issue of unbalanced variances on different directions.
4) AGH generates the graph representation of each data point, and then uses the resulting new representations to encode the testing data points with binary codes.
4) IMH first employs any non-parametric dimensionality reduction methods to convert original high-dimensional data into their low-dimensional feature spaces via preserving the local similarity structures of the data, and then proposes a heuristic method to learn the hash functions.
5) LLH first captures the manifold structures of original data using locality-sensitive sparse coding, and then designs a joint minimization problem of the embedding error and the quantization loss to recover the resulting manifold structures in a low-dimensional Hamming space.

B. Data Sets

The CIFAR data set consists of 60,000 color images from 10 classes, where each class contains 6,000 images and each image is represented by a 512-dimensional GIST feature vector. In our experiments, we select 50,000 images (i.e., 5,000 images per class) to be the representative set for hash function learning in training stage, while the left 10,000 images are testing set.

The MNIST data set consists of 70,000 images which are associated with a digit from '0' to '9'. We represented each image with a 784-dimensional feature vector. Following the setting in [31], [33], we split MNIST into two parts, i.e., training set and testing set, respectively, contains 69,000 images and 1,000 images.

The NUS-WIDE data set in our experiments consists of 195,969 images, in which each image is annotated by at least one of the 21 most frequent labels out of original 81 concept tags. Each image is represented by a 500-dimensional SIFT feature vector. We uniformly sample 100 images from each of these 21 tags to form testing set of 2,100 images. The left 193,869 images serve as the big data set.

The GIST data set includes 500,000 images and 1,000 testing images, where each image is represented by a 960-dimensional global GIST feature vector.

C. Evaluation Metric

We conduct two similarity search by following previous literatures [21], [31], i.e., hash lookup and Hamming ranking, respectively. Hash lookup needs constant time complexity with a single hash table, while Hamming ranking measures the search quality via ranking the retrieved data points in terms of the Hamming distances to a specific query.

In our experiments, we first use the mean precision of HAMming radius 2 (HAM2) and the Mean Average Precision (MAP), respectively, to evaluate the results of hash lookup and Hamming ranking under different hash bits. We also evaluate all the hashing methods in terms of precision-recall curves and time cost.

By regarding the evaluation metric HAM2, we set the threshold of Hamming radius as 2 to output the retrieved training data, whose Hamming distance is less than 2 to the query. We then report the mean precision of all the queries.

Given a query and *r* retrieved data points by the hashing methods, we define MAP as follows:

$$AP(q) = \frac{1}{l}\sum_{r=1}^{R} P(r)\delta(r)$$

$$\mathrm{MAP} = \frac{1}{q}\sum_{q=1}^{Q} AP(q) \tag{18}$$

where l and $P(r)$, respectively, are the number of true neighbors of the ground truth and the precision of the top r retrieved training data, and $\delta(r) = 0$ means the *r*-th retrieved data point is a false neighbor of the query, otherwise $\delta(r) = 1$. MAP is thus defined as the mean of all queries' average precision. Clearly, the larger the HAM2 or MAP, the better the hashing performance is.

The data sets (such as CIFAR and MNIST) have class labels for each image, so we regard their true nearest neighbors as the semantic nearest neighbors, i.e., the images will be regarded as neighbors if they share the same digital labels. For the data sets (such as NUS-WIDE and GIST), we follow the literatures [29], [31] to replace MAP with Mean Precision (MP) of top-5000 returned neighbors since the calculation of MAP is time consuming on the big data set.

D. Parameters Settings

For fair comparison, we set all the training data as the representative set on the data sets CIFAR and MINST, while we uniformly select 63,000 (i.e., 300 images per class) and 60, 000, respectively, for the data sets NUS-WIDE and GIST, as the representative set, to learn the hash functions. We set the number of landmarks as $\{100, 300, 500, 700, 900\}$ for the methods (such

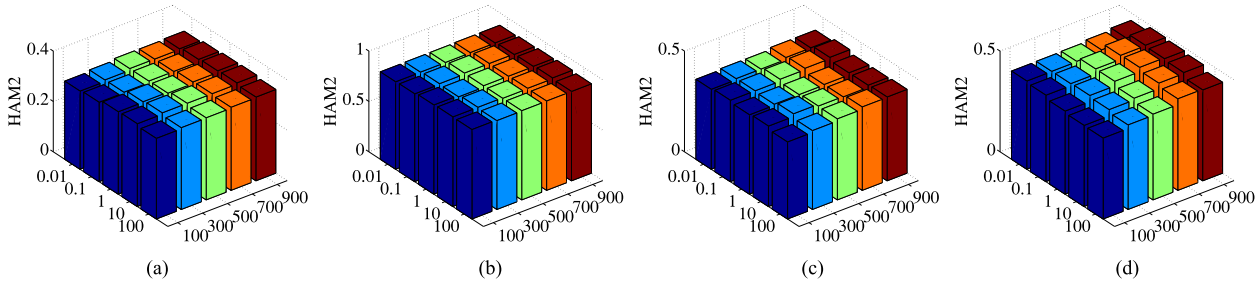

Fig. 1. HAM2 results of our proposed gPCA with different numbers of landmarks and different values of α in (10) while setting the code length as 32, on four data sets. (a) CIFAR. (b) MNIST. (c) NUS-WIDE. (d) GIST.

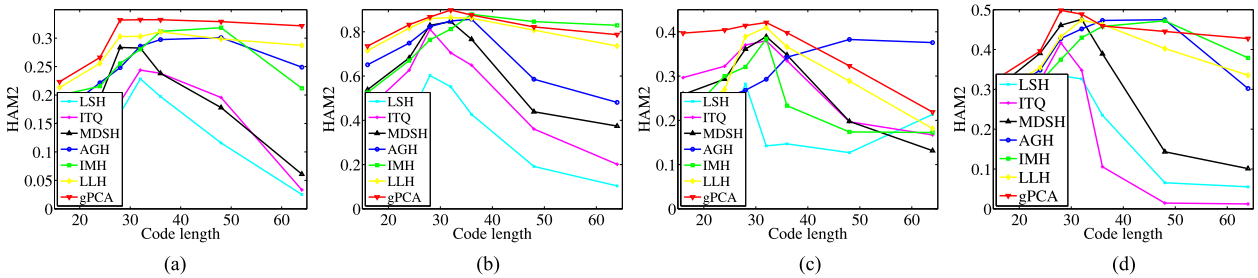

Fig. 2. HAM2 results of all hashing methods on four data sets at different number of hash bits, i.e., $c \in [12, 16, 24, 28, 32, 48, 64]$. (a) CIFAR. (b) MNIST. (c) NUS-WIDE. (d) GIST.

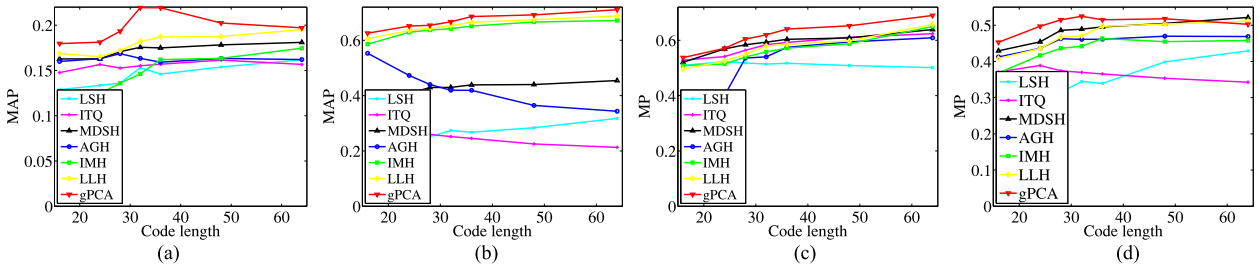

Fig. 3. Results of MAP and MP, respectively, of all hashing methods, on the data sets CIFAR and MNIST and the data sets NUS-WIDE and GIST, at different number of hash bits, i.e., $c \in [12, 16, 24, 28, 32, 48, 64]$. (a) CIFAR. (b) MNIST. (c) NUS-WIDE. (d) GIST.

as AGH, IMH, and our gPCA) using k-means to generate the landmarks, and also set the range of all the parameters of all the hashing methods as $[0.01, 0.1, 1, 10, 100]$, where all the methods output their best hashing performance.

E. Result Analysis

We report the parameters' sensitivity on different numbers of the landmarks and the variation of α in (10) in Fig. 1, and list the results of three evaluation metrics of all hashing methods in Figs. 2–4. We also report the running time of all the hashing methods in Table III.

From Fig. 1, we know that our method achieves better hashing results while increasing the number of landmarks. The reason is that our proposed gPCA method may easier estimate the distribution of the data set with more landmarks. On the other hand, our proposed method selects different values of α at different data sets to achieve the best hashing performanc. Specially, the data sets (such as CIFAR and MNIST) select large value (e.g., $\alpha \in [10, 100]$) and the other two select small value (e.g., $\alpha \in [0.1, 1]$). It is noteworthy that the parameter α in (10) is used to balance the weight between the PCA part and the manifold part, i.e., the large value of α enables to add the weight of the PCA part. According to our experimental results, we know that small data sets (such as CIFAR and MNIST) need more weight of local structures of the data than large data sets (such as NUS-WIDE and GIST).

By regarding Figs. 2–4, our proposed gPCA method achieves the best hashing performance and has comparable running time to state-of-the-arts hashing methods. For example, our method on average improves by 42.2%, 20.2%, 8.6%, and 5.7%, respectively, than the data-independent hashing method (i.e., LSH), the best PCA hashing method (i.e., MDSH), the best Manifold based hashing method (i.e., IMH), and the best comparison method (i.e., LLH), in terms of the HAM2 results in Fig. 2. We list more observations as follows:

1) Data-dependent hashing methods (i.e., PCA based methods, Manifold based methods, LLH, and our gPCA) outperform data-independent method, i.e., LSH. For example, data-dependent hashing methods on average

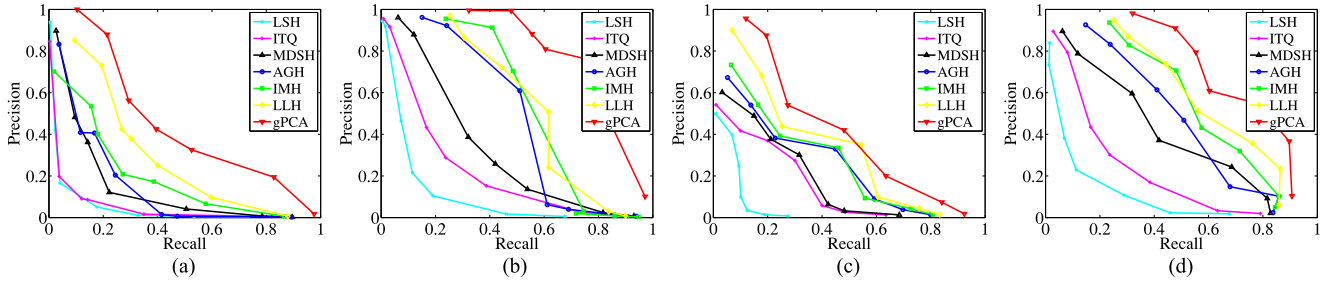

Fig. 4. Precision-recall curves of all hashing methods, on four data sets at different number of hash bits, i.e., $c \in [12, 16, 24, 28, 32, 48, 64]$. (a) CIFAR. (b) MNIST. (c) NUS-WIDE. (d) GIST.

TABLE III
RUNNING TIME (RECORDED IN SECONDS) FOR ALL THE HASHING METHODS

Methods	CIFAR		MNIST		NUSWIDE		GIST	
	training	testing	training	testing	training	testing	training	testing
LSH	**1.1**	1.5×10^{-5}	**1.3**	2.3×10^{-5}	**3.5**	1.8×10^{-5}	**20.1**	3.7×10^{-4}
ITQ	14.5	4.7×10^{-5}	15.0	2.5×10^{-5}	30.4	2.1×10^{-5}	135.0	4.5×10^{-4}
MDSH	10.5	4.5×10^{-5}	8.3	5.1×10^{-5}	26.1	1.9×10^{-5}	100.1	6.9×10^{-4}
AGH	19.6	8.4×10^{-5}	19.0	3.6×10^{-5}	26.3	5.6×10^{-5}	163.3	4.5×10^{-4}
IMH	17.2	5.2×10^{-5}	20.1	4.8×10^{-5}	20.2	4.7×10^{-5}	166.4	5.2×10^{-4}
LLH	22.2	9.7×10^{-5}	32.6	5.5×10^{-5}	56.2	9.9×10^{-5}	186.5	6.5×10^{-4}
gPCA	15.9	5.1×10^{-5}	16.0	4.3×10^{-5}	29.2	5.0×10^{-5}	150.2	5.8×10^{-4}

improve by more than 32%, compared to the LSH method. This indicates that it is reasonable for taking the data distribution into account for conducting similarity search. Actually, we can regard the learnt data distribution as prior knowledge. That is, data-dependent hashing methods consider constructing hashing with prior knowledge, while data-independent hashing method LSH does not take it into account.

2) We find that the methods (such as LLH and our gPCA) on average improve by about 8.26% and 22.85%, respectively, compared to manifold based hashing methods and PCA based hashingn methods, since both LLH and our gPCA simultaneously consider preserving the local and global similarity structures of the data. This implies that preserving two kinds of similarity structures enables to achieve better hashing performance, compared to the methods which only preserve each of these similarity structures in our experiments. Our experimental results further demonstrate that these two kinds of similarity structures provide complimentary information to each other, as shown in previous literatures [58], [64].
3) Our gPCA method outperforms LLH in terms of all three evaluation metrics since the sequential steps in LLH easily leads to suboptimal.
4) In terms of training time, data-dependent hashing methods (such as ITQ, MDSH, AGH, IMH, LLH, and gPCA) are more time consuming than data-independent hashing methods (i.e., LSH) since the former methods need to conduct a learning process using training data. More specifically, LSH is the fastest one, ITQ, MDSH, AGH, IMH, LLH and gPCA have similar training time. In terms of testing time, all the hashing methods are in the same magnitude.

VI. CONCLUSION

In this paper, we proposed a novel and effective unsupervised uni-modal hashing method via preserving the local and global similarity structures of the data. To do this, we first partitioned the whole data set into groups (or clusters) and generated the probability representations of all the data points via sparsely representing them by the landmarks. We then devised a new objective function to conduct an eigenvalue problem on the representative set, and further proved that such a method preserves the neighborhood of the data. Our experimental results on public benchmark data sets showed that our proposed gPCA hashing method leaded to very competitive hashing performance, compared to the comparison hashing methods, and achieved real-time similarity retrieval in big data. In this paper, we only considered the linear relationship among the data set and thus cannot model the complex relationship among the big data set. In our future work, we will extend our proposed framework to consider the nonlinear relationship among the data via kernel trick.

REFERENCES

[1] X. Mao, Y. Yang, and N. Li, "Hashing with pairwise correlation learning and reconstruction," *IEEE Trans. Multimedia*, vol. 19, no. 2, pp. 382–392, Feb. 2017.
[2] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen, "Inter-media hashing for large-scale retrieval from heterogeneous data sources," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 785–796.
[3] Y. Hao *et al.*, "Stochastic multiview hashing for large-scale near-duplicate video retrieval," *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 1–14, Jan. 2017.
[4] X. Nie, Y. Chai, J. Liu, J. Sun, and Y. Yin, "Spherical torus-based video hashing for near-duplicate video detection," *Sci. China Inf. Sci.*, vol. 59, no. 5, 2016, Art. no. 059101.
[5] Q. Wen, D. Wang, S. Feng, Y. Zhang, and G. Yu, "A novel cross-modal hashing algorithm based on multimodal deep learning," *Sci. China Inf. Sci.*, vol. 60, no. 9, 2017, Art. no. 092104.
[6] Z. Chen, J. Lu, J. Feng, and J. Zhou, "Nonlinear discrete hashing," *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 123–135, Jan. 2017.
[7] X. Lu, X. Zheng, and X. Li, "Latent semantic minimal hashing for image retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 355–368, Jan. 2017.

[8] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1606.00185

[9] X. Zhu, Z. Huang, J. Cui, and H. T. Shen, "Video-to-shot tag propagation by graph sparse group lasso," *IEEE Trans. Multimedia*, vol. 15, no. 3, pp. 633–646, Apr. 2013.

[10] X. Peng, Z. Yu, Z. Yi, and H. Tang, "Constructing the l2-graph for robust subspace learning and subspace clustering," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1053–1066, Apr. 2017.

[11] X. Li, D. Hu, and F. Nie, "Large graph hashing with spectral rotation," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 2203–2209.

[12] J. Song, L. Gao, Y. Yan, D. Zhang, and N. Sebe, "Supervised hashing with pseudo labels for scalable multimedia retrieval," in *Proc. ACM MM*, 2015, pp. 827–830.

[13] X. Zhu, Z. Huang, H. T. Shen, and X. Zhao, "Linear cross-modal hashing for efficient multimedia search," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2013, pp. 143–152.

[14] H. Jégou *et al.*, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.

[15] R. Hu *et al.*, "Graph self-representation method for unsupervised feature selection," *Neurocomputing*, vol. 220, pp. 130–137, 2017.

[16] Y. Zhen, Y. Gao, D. Yeung, H. Zha, and X. Li, "Spectral multimodal hashing and its application to multimedia retrieval," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 27–38, Jan. 2016.

[17] H. Jain, P. Pérez, R. Gribonval, J. Zepeda, and H. Jégou, "Approximate search with quantized sparse representations," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 681–696.

[18] R. Ye and X. Li, "Compact structure hashing via sparse and similarity preserving embedding," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 718–729, Mar. 2016.

[19] X. Zhu, K. Thung, J. Zhang, and D. Shen, "Fast neuroimaging-based retrieval for Alzheimer's disease analysis," in *Proc. MLMI*, 2016, pp. 313–321.

[20] 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 *Proc. 19th ACM Int. Conf. Multimedia*, 2011, pp. 423–432.

[21] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen, "Sparse hashing for fast multimedia search," *ACM Trans. Inf. Syst.*, vol. 31, no. 2, May 2013, Art. no. 9.

[22] X. Peng, H. Tang, L. Zhang, Z. Yi, and S. Xiao, "A unified framework for representation-based subspace clustering of out-of-sample and large-scale data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2499–2512, Dec. 2016.

[23] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geom.*, 2004, pp. 253–262.

[24] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.

[25] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep.-Oct. 2009, pp. 2130–2137.

[26] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1509–1517.

[27] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.

[28] Y. Weiss, R. Fergus, and A. Torralba, "Multidimensional spectral hashing," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 340–353.

[29] 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.*, vol. 35, no. 12, pp. 2916–2929, Dec, 2013.

[30] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.

[31] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1–8.

[32] F. Shen *et al.*, "Hashing on nonlinear manifolds," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1839–1851, Jun. 2015.

[33] X. Zhu, L. Zhang, and Z. Huang, "A sparse embedding and least variance encoding approach to hashing," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3737–3750, Sep. 2014.

[34] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu, "Semi-supervised nonlinear hashing using bootstrap sequential projection learning," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1380–1393, Jun. 2013.

[35] G. Irie, Z. Li, X. Wu, and S. Chang, "Locally linear hashing for extracting non-linear manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 2123–2130.

[36] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 37–45.

[37] S. Conjeti *et al.*, "Metric hashing forests," *Med. Image Anal.*, vol. 34, pp. 13–29, 2016.

[38] X. Zhu, H. Suk, S. Lee, and D. Shen, "Subspace regularized sparse multitask learning for multiclass neurodegenerative disease identification," *IEEE Trans. Biomed. Eng.*, vol. 63, no. 3, pp. 607–618, Mar. 2016.

[39] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 1711–1717.

[40] W. Kong and W.-J. Li, "Double-bit quantization for hashing," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, vol. 1, no. 2, pp. 634–640.

[41] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to hash for indexing big data a survey," *Proc. IEEE*, vol. 104, no. 1, pp. 34–57, Jan. 2016.

[42] S. Zhang and D. Metaxas, "Large-scale medical image analytics: Recent methodologies, applications and future directions," *Med. Image Anal.*, vol. 33, pp. 98–101, 2016.

[43] T.-T. Do, A.-D. Doan, and N.-M. Cheung, "Learning to hash with binary deep neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 219–234.

[44] J. Heo, Y. Lee, J. He, S. Chang, and S. Yoon, "Spherical hashing: Binary code embedding with hyperspheres," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2304–2316, Nov. 2015.

[45] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, vol. 22, pp. 1042–1050.

[46] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 353–360.

[47] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Commun. ACM*, vol. 51, no. 1, pp. 117–122, 2008.

[48] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2008, pp. 1–8.

[49] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 2475–2483.

[50] X. Peng, J. Lu, Z. Yi, and Y. Rui, "Automatic subspace learning via principal coefficients embedding," *IEEE Trans. Cybern.*, to be published.

[51] Y. Wei, Y. Song, Y. Zhen, B. Liu, and Q. Yang, "Heterogeneous translated hashing: A scalable solution towards multi-modal similarity search," *ACM Trans. Knowl. Discovery Data*, vol. 10, no. 4, 2016, Art. no. 36.

[52] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing." in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, vol. 1, no. 2, p. 5.

[53] J. Zhou, G. Ding, and Y. Guo, "Latent semantic sparse hashing for cross-modal similarity search," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 415–424.

[54] Q.-Y. Jiang and W.-J. Li, "Deep cross-modal hashing," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1602.02255

[55] X. Zhu, X. Li, S. Zhang, C. Ju, and X. Wu, "Robust joint graph sparse coding for unsupervised spectral feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1263–1275, Jun. 2017.

[56] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering." in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, vol. 14, pp. 585–591.

[57] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2010, pp. 18–25.

[58] X. Zhu, X. Li, and S. Zhang, "Block-row sparse multiview multilabel learning for image classification," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 450–461, Feb. 2016.

[59] D. Cai and X. Chen, "Large scale spectral clustering via landmark-based sparse representation," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1669–1680, Aug. 2015.

[60] D. Zhang, F. Wang, and L. Si, "Composite hashing with multiple information sources," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2011, pp. 225–234.

[61] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.
[62] J. Goldberger, S. T. Roweis, G. E. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 513–520.
[63] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 106.
[64] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[65] A. Andoni, T. Laarhoven, I. Razenshteyn, and E. Waingarten, "Optimal hashing-based time-space trade-offs for approximate near neighbors," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1608.03580
[66] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.
[67] T.-S. Chua *et al.*, "Nus-wide: A real-world web image database from National University of Singapore," in *Proc. ACM Int. Conf. Image Video Retrieval*, 2009, Art. no. 48.