Latent Structure Preserving Hashing

Ziyun Cai[1]
cziyun1@sheffield.ac.uk
Li Liu[2]
li2.liu@northumbria.ac.uk
Mengyang Yu[2]
m.y.yu@ieee.org
Ling Shao[2]
ling.shao@ieee.org

[1] Department of Electronic and Electrical Engineering
The University of Sheffield
Sheffield, S1 3JD, UK

[2] Computer Vision and Artificial Intelligence Group
Department of Computer Science and Digital Technologies
Northumbria University
Newcastle upon Tyne, NE1 8ST, UK

Abstract

Aiming at efficient similarity search, hash functions are designed to embed high-dimensional feature descriptors to low-dimensional binary codes such that similar descriptors will lead to the binary codes with a short distance in the Hamming space. It is critical to effectively maintain the intrinsic structure and preserve the original information of data in a hashing algorithm. In this paper, we propose a novel hashing algorithm called Latent Structure Preserving Hashing (LSPH), with the target of finding a well-structured low-dimensional data representation from the original high-dimensional data through a novel objective function based on Nonnegative Matrix Factorization (NMF). Via exploiting the probabilistic distribution of data, LSPH can automatically learn the latent information and successfully preserve the structure of high-dimensional data. After finding the low-dimensional representations, the hash functions can be acquired through multi-variable logistic regression. Experimental results on two large-scale datasets, i.e., **SIFT 1M** and **GIST 1M**, show that LSPH can significantly outperform the state-of-the-art hashing techniques.

1 Introduction

Similarity search [3, 11, 19, 22] is one of the most critical problems in information retrieval as well as in pattern recognition, data mining and machine learning. Generally speaking, effective similarity search approaches try to construct the index structure in the metric space. However, with the increase of the dimensionality of the data, how to implement the similarity search efficiently and effectively has become an significant topic. To improve retrieval efficiency, hashing algorithms are deployed to find a hash function from Euclidean space to Hamming space. The hashing algorithms with binary coding techniques mainly have two advantages: (1) binary hash codes save storage space; (2) it is efficient to compute the Hamming distance (XOR operation) between the training data and the new coming data in the retrieval procedure of similarity search. The time complexity of searching the hashing table is near $O(1)$.

Pages 172.1-172.11
DOI: https://dx.doi.org/10.5244/C.29.172

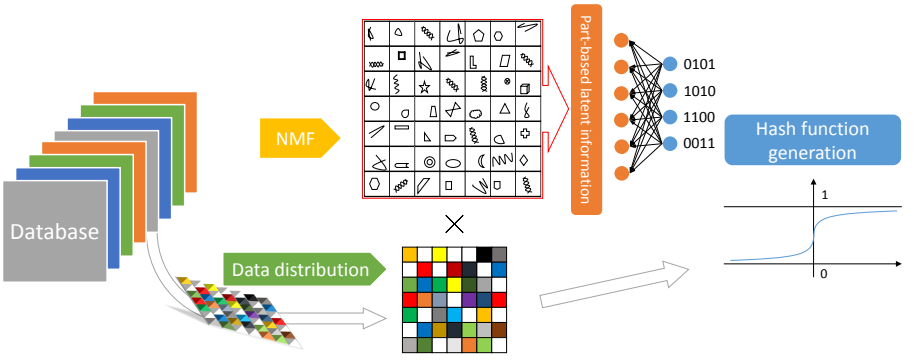

Figure 1: The outline of the proposed method. Part-based latent information is learned from NMF with the regularization of data distribution.

Current hashing algorithms can be roughly divided into random projection based or learning based. Locality Sensitive Hashing (LSH) [3] is one of the most widely used hashing algorithms based on random linear projections, which can efficiently map the data points from the high-dimensional space into the low-dimensional Hamming space. Kernelized Locality Sensitive Hashing (KLSH) [7] can exploit more significant similarity in kernel space for better retrieval effectiveness. However, the random projection based hash functions are effective only when the binary hash code is long enough. Through mining the structure of the data, then being represented on the objective function, a learning based hashing algorithm can obtain the hash function by solving an optimization problem associated with the objective function. Spectral Hashing (SpH) [20] is a representative unsupervised hashing method, which can learn compact binary codes that preserve the similarity between documents by forcing the balanced and uncorrelated constraints into the learned codes. In [18], Principal Component Analysis Hashing (PCAH) has been introduced for better quantization rather than random projection hashing. Moreover, Semantic Hashing (SH), which is based on Restricted Boltzmann Machines (RBM) [16], was proposed in [15]. Liu et al. [12] proposed an Anchor Graph-based Hashing method (AGH), which automatically discovers the neighborhood structure inherent in the data to learn appropriate compact codes. More recently, Spherical Hashing (SpherH) [5] and Iterative Quantization (ITQ) [4] were developed for more compact and effective binary coding. Additionally, Boosted Similarity Sensitive Coding (BSSC) [17], Compressed Hashing (CH) [10] and Self-taught Hashing (STH) [22] are also successfully utilized for large-scale visual retrieval tasks.

However, the above mentioned hashing methods have their limitations. Though the random projection based hashing methods can produce compact codes, the simple, linear hash functions cannot reflect the underlying relationship between the data points. Meanwhile, since the linear formulation is computed with a high-dimensional matrix, it will result in a heavy computational cost. On the other hand, the learning based hashing algorithms are not efficient when the codewords are long. Besides, those hashing approaches, which first reduce the dimensionality of the original data, cannot obtain a well-structured low-dimensional data representation.

In order to overcome these limitations, we propose a novel NMF-based approach called Latent Structure Preserving Hashing (LSPH) which can effectively preserve data probabilis-

tic distribution and capture much of the locality structure from the high-dimensional data. Moveover, the nonnegative matrix factorization can automatically learn the latent information and the part-based representations of data. Incorporated with the representation of binary codes, the part-based latent information obtained by NMF could be regarded as independent latent attributes of samples. In other words, the binary codes determine whether a sample possesses the corresponding latent attributes.

2 Latent Structure Preserving Hashing

In this section, we mainly explain the proposed Latent Structure Preserving Hashing algorithm. Firstly, because of the limitation of NMF, which cannot completely discover the locality structure of the original high-dimensional data, we provide a new objective function to preserve as much of the probabilistic distribution structure of the high-dimensional data as possible to the low-dimensional map. Meanwhile, we propose an optimization framework for the objective function and show the updating rules. Secondly, to implement the optimization process, the training data are relaxed to a real-valued range. Then, we convert the real-valued representations into binary codes. Finally, we analyze the experimental results and compare them with several existing hashing algorithms. The outline of the proposed LSPH approach is depicted in Fig. 1.

2.1 Preserving Data Structure with NMF

NMF is an unsupervised learning algorithm which can learn a parts-based representation. Theoretically, it is expected that the low-dimensional data V given by NMF can obtain locality structure from the high-dimensional data X. However, in real-world applications, NMF cannot discover the intrinsic geometrical and discriminating structure of the data space. Therefore, to preserve as much of the significant structure of the high-dimensional data as possible, we propose to minimize the Kullback-Leibler divergence between the joint probability distribution in the high-dimensional space and the joint probability distribution that is heavy-tailed in the low-dimensional space:

$$C = \lambda KL(P\|Q). \tag{1}$$

In Eq. (1), P is the joint probability distribution in the high-dimensional space which can also be denoted as p_{ij}. Q is the joint probability distribution in the low-dimensional space that can be represented as q_{ij}. λ is the control of the smoothness of the new representation. The conditional probability p_{ij} means the similarity between data points $\mathbf{x}_i$ and $\mathbf{x}_j$, where $\mathbf{x}_j$ is picked in proportion to their probability density under a Gaussian centered at $\mathbf{x}_i$. Since only significant points are needed to model pairwise similarities, we set p_{ii} and q_{ii} to zero. Meanwhile, it has the characteristics that $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$ for $\forall i, j$. The pairwise similarities in the high-dimensional space p_{ij} are defined as:

$$p_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq l}\exp(-\|\mathbf{x}_k - \mathbf{x}_l\|^2/2\sigma_k^2)}, \tag{2}$$

where σ_i is the variance of the Gaussian distribution which is centered on data point x_i. Each data point x_i makes a significant contribution to the cost function. In the low-dimensional

map, using the probability distribution that is heavy tailed, the joint probabilities q_{ij} can be defined as:

$$q_{ij} = \frac{(1+\|\mathbf{v}_i - \mathbf{v}_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|\mathbf{v}_k - \mathbf{v}_l\|^2)^{-1}}. \tag{3}$$

This definition is an infinite mixture of Gaussians, which is much faster to evaluate the density of a point than the single Gaussian, since it does not have an exponential. This representation also makes the mapped points invariant to the changes in the scale for the embedded points that are far apart. Thus, the cost function based on Kullback-Leibler divergence can effectively measure the significance of the data distribution . q_{ij} models p_{ij} is given by

$$G = KL(P\|Q) = \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij}. \tag{4}$$

For simplicity, we define two auxiliary variables d_{ij} and Z for making the derivation clearer as follows:

$$d_{ij} = \|\mathbf{v}_i - \mathbf{v}_j\| \text{ and } Z = \sum_{k\neq l}(1+d_{kl}^2)^{-1}. \tag{5}$$

Therefore, the gradient of function G with respect to $\mathbf{v}_i$ can be given by

$$\frac{\partial G}{\partial \mathbf{v}_i} = 2\sum_{j=1}^{N} \frac{\partial G}{\partial d_{ij}}(\mathbf{v}_i - \mathbf{v}_j). \tag{6}$$

Then $\frac{\partial G}{\partial d_{ij}}$ can be calculated by Kullback-Leibler divergence in Eq. (4):

$$\frac{\partial G}{\partial d_{ij}} = -\sum_{k\neq l} p_{kl}\left(\frac{1}{q_{kl}Z}\frac{\partial((1+d_{kl}^2)^{-1})}{\partial d_{ij}} - \frac{1}{Z}\frac{\partial Z}{\partial d_{ij}}\right). \tag{7}$$

Since $\frac{\partial((1+d_{kl}^2)^{-1})}{\partial d_{ij}}$ is nonzero if and only if $k=i$ and $l=j$, and $\sum_{k\neq l} p_{kl} = 1$, the gradient function can be expressed as

$$\frac{\partial G}{\partial d_{ij}} = 2(p_{ij} - q_{ij})(1+d_{ij}^2)^{-1}. \tag{8}$$

Eq. (8) can be substituted into Eq. (6). Therefore, the gradient of the Kullback-Leibler divergence between P and Q is

$$\frac{\partial G}{\partial \mathbf{v}_i} = 4\sum_{j=1}^{N}(p_{ij} - q_{ij})(\mathbf{v}_i - \mathbf{v}_j)(1+\|\mathbf{v}_i - \mathbf{v}_j\|^2)^{-1}. \tag{9}$$

Therefore, through combining the data structure preserving part in Eq. (1) and the NMF technique, we can obtain the following new objective function:

$$O_f = \|X - UV\|^2 + \lambda KL(P\|Q), \tag{10}$$

where $V \in \{0,1\}^{D\times N}$, $X, U, V \geqslant 0$, $U \in \mathbb{R}^{M\times D}$, $X \in \mathbb{R}^{M\times N}$, and λ controls the smoothness of the new representation.

In most of the circumstances, the low-dimensional data only from NMF is not effective and meaningful for realistic applications. Thus, we introduce $\lambda KL(P\|Q)$ to preserve the structure of the original data which can obtain better results in information retrieval.

2.2 Relaxation and Optimization

Since the discreteness condition $V \in \{0,1\}^{D\times N}$ in Eq. (10) cannot be calculated directly in the optimization procedure, motivated by [20], we first relax the data $V \in \{0,1\}^{D\times N}$ to the range $V \in \mathbb{R}^{D\times N}$ for obtaining real-values. Then let the Lagrangian of our problem be:

$$\mathcal{L} = \|X - UV\|^2 + \lambda KL(P\|Q) + tr(\Phi U^T) + tr(\Psi V^T), \tag{11}$$

where matrices Φ and Ψ are two Lagrangian multiplier matrices. Since we have the gradient of $C = \lambda G$:

$$\frac{\partial C}{\partial \mathbf{v}_i} = 4\lambda \sum_{j=1}^{N} (p_{ij} - q_{ij})(\mathbf{v}_i - \mathbf{v}_j)(1 + \|\mathbf{v}_i - \mathbf{v}_j\|^2)^{-1}, \tag{12}$$

we make the gradients of $\mathcal{L}$ be zeros to minimize O_f:

$$\frac{\partial \mathcal{L}}{\partial V} = 2(-U^T X + U^T UV) + \frac{\partial C}{\partial \mathbf{v}_i} + \Psi = \mathbf{0}, \tag{13}$$

$$\frac{\partial \mathcal{L}}{\partial U} = 2(-XV^T + UVV^T) + \Phi = \mathbf{0}, \tag{14}$$

In addition, we also have KKT conditions: $\Phi_{ij}U_{ij} = 0$ and $\Psi_{ij}V_{ij} = 0, \forall i, j$. Then multiplying V_{ij} and U_{ij} in the corresponding positions on both sides of Eqs. (13) and (14) respectively, we obtain

$$(2(-U^T X + U^T UV) + \frac{\partial C}{\partial \mathbf{v}_i})_{ij} V_{ij} = 0, \tag{15}$$

$$2(-XV^T + UVV^T)_{ij} U_{ij} = 0. \tag{16}$$

Note that

$$\left(\frac{\partial C}{\partial \mathbf{v}_j}\right)_i = \left(4\lambda \sum_{k=1}^{N} \frac{p_{jk}\mathbf{v}_j - q_{jk}\mathbf{v}_j - p_{jk}\mathbf{v}_k + q_{jk}\mathbf{v}_k}{1 + \|\mathbf{v}_j - \mathbf{v}_k\|^2}\right)_i$$
$$= 4\lambda \sum_{k=1}^{N} \frac{p_{jk}V_{ij} - q_{jk}V_{ij} - p_{jk}V_{ik} + q_{jk}V_{ik}}{1 + \|\mathbf{v}_j - \mathbf{v}_k\|^2}.$$

Therefore, we have the following update rules for any i, j:

$$V_{ij} \leftarrow \frac{(U^T X)_{ij} + 2\lambda \sum_{k=1}^{N} \frac{p_{jk}V_{ik} + q_{jk}V_{ij}}{1+\|\mathbf{v}_j - \mathbf{v}_k\|^2}}{(U^T UV)_{ij} + 2\lambda \sum_{k=1}^{N} \frac{p_{jk}V_{ij} + q_{jk}V_{ik}}{1+\|\mathbf{v}_j - \mathbf{v}_k\|^2}} V_{ij}, \tag{17}$$

$$U_{ij} \leftarrow \frac{(XV^T)_{ij}}{(UVV^T)_{ij}} U_{ij}. \tag{18}$$

All the elements in U and V can be guaranteed that they are nonnegative from the allocation. In [8], it has been proved that the objective function is monotonically non-increasing after each update of U or V. The proof of convergence about U and V is similar to the ones in [2, 23].

2.3 Hash Function Generation

The low-dimensional representations $V \in \mathbb{R}^{D\times N}$ and the bases $U \in \mathbb{R}^{M\times D}$, where $D \ll M$, can be obtained from Eq. (17) and Eq. (18), respectively. Then we need to convert the low-dimensional real-valued representations from $V = [\mathbf{v}_1, \cdots, \mathbf{v}_N]$ into binary codes via thresholding: if the d-th element in $\mathbf{v}_n$ is larger than a specified threshold, this real value will be represented as 1; otherwise it will be 0, where $d = 1, \cdots, D$ and $n = 1, \cdots, N$.

In addition, a well-designed semantic hashing should also be entropy maximizing to ensure its efficiency [1]. Meanwhile, from the information theory, through having a uniform probability distribution, the source alphabet can reach a maximal entropy. Specifically, if the entropy of codes over the corpus is small, the documents will be mapped to a small number of codes (hash bins). In this paper, the threshold of the elements in $\mathbf{v}_n$ can be set to the median value of $\mathbf{v}_n$, which can satisfy entropy maximization. Therefore, half of the bit-strings will be 1 and the other half will be 0. In this way, the real-value code can be calculated into a binary code [21].

However, from the above procedure, we can only obtain the binary codes of the data in the training set. Therefore, given a new sample, it cannot directly find a hash function. In our approach, due to the binary code environment, we use the logistic regression [6] which can be treated as a type of probabilistic statistical classification model to compute the hash code in the test set. Before obtaining the logistic regression cost function, we define that the binary code is represented as $\hat{\mathbf{V}} = [\hat{\mathbf{v}}_1, \cdots, \hat{\mathbf{v}}_N]$, where $\hat{\mathbf{v}}_n \in \{0,1\}^D$ and $n = 1, \cdots, N$. Therefore, the training set can be considered as $\{(\mathbf{v}_1, \hat{\mathbf{v}}_1), (\mathbf{v}_2, \hat{\mathbf{v}}_2), \cdots, (\mathbf{v}_N, \hat{\mathbf{v}}_N)\}$. The vector-valued regression function which is based on the corresponding regression matrix $\Theta \in \mathbb{R}^{D\times D}$ can be represented as

$$h_\Theta(\mathbf{v}_n) = \left(\frac{1}{1 + e^{-(\Theta^T \mathbf{v}_n)_i}} \right)^T_{i=1,\cdots,D}. \tag{19}$$

Therefore, with the maximum log-likelihood criterion for the Bernoulli-distributed data, our cost function for the corresponding regression matrix can be defined as:

$$J(\Theta) = -\frac{1}{N} \left(\sum_{n=1}^{N} \left(\hat{\mathbf{v}}_n^T \mathbf{log}(h_\Theta(\mathbf{v}_n)) + (\mathbf{1} - \hat{\mathbf{v}}_n)^T \mathbf{log}(\mathbf{1} - h_\Theta(\mathbf{v}_n)) \right) + \delta \|\Theta\|^2 \right), \tag{20}$$

where $\mathbf{log}(\cdot)$ is the element-wise logarithm function and $\mathbf{1}$ is an $D \times 1$ all ones matrix. We use $\delta\|\Theta\|^2$ as the regularization term in logistic regression to avoid overfitting.

To find the matrix Θ which aims to minimize $J(\Theta)$, we use gradient descent and repeatedly update each parameter using a learning rate α. The updating equation is shown as follows:

$$\Theta^{(t+1)} = \Theta^{(t)} - \frac{\alpha}{N} \sum_{n=1}^{N} (h_{\Theta^{(t)}}(\mathbf{v}_n) - \hat{\mathbf{v}}_n) \mathbf{v}_n^T - \frac{\alpha\delta}{N} \Theta^{(t)}. \tag{21}$$

The updating equation stops when the norm of difference between $\Theta^{(t+1)}$ and $\Theta^{(t)}$, i.e., $||\Theta^{(t+1)} - \Theta^{(t)}||^2$, is smaller than a small value. Then we can obtain the regression matrix Θ.

After this, we can obtain the real-valued low-dimensional representation through a linear projection matrix $Q = (U^T U)^{-1} U^T$, which is the pseudoinverse of U. Since we have $X \approx UV$, applying the projection matrix Q to the data matrix X, we obtain $QX = (U^T U)^{-1} U^T X \approx (U^T U)^{-1} U^T UV = V$. Note that each entry of h_Θ is a sigmoid function, the hash codes for a new coming sample $X_{new} \in \mathbb{R}^{M\times 1}$ can be represented as:

$$\hat{V}_{new} = \lfloor h_\Theta(QX_{new}) \rceil, \tag{22}$$

where $\lfloor \cdot \rceil$ means the nearest integer function for each entry of h_Θ. We define that the threshold of binarization is 0.5. Therefore, if a bit from $h_\Theta(QX_{new})$ is larger than 0.5, it will be represented as 1, otherwise 0. For example, through Eq. (22), vector [0.17, 0.37, 0.50, 0.79, 0.03, 0.92, $\cdots$] is expressed as [0, 0, 0, 1, 0, 1, $\cdots$]. Up to now, we can obtain the Latent Structure Preserving Hashing codes for both training and test data. The procedure of LSPH is summarized in Algorithm 1.

Algorithm 1 Latent Structure Preserving Hashing (LSPH)

Input:

The training matrix $X \in \mathbb{R}^{M \times N}$; the objective dimension (code length) D of hash codes; the learning rate α for logistic regression; the regularization parameters $\{\delta, \lambda\}$.

Output:

The basis matrix U and the regression matrix Θ.

1: **repeat**
2: Compute the low-dimensional representation matrix V and the basis matrix U via Eq. (17) and Eq. (18);
3: **until** convergence
4: Obtain the regression matrix Θ through Eq. (21) and the final LSPH encoding for each sample is defined in Eq. (22).

2.4 Computational Complexity Analysis

In this section, we will discuss the computational complexity of our LSPH, which consists of three parts. The first part is for computing NMF, the complexity of which is $O(NMKD)$ [9], where N is the size of the dataset, M and D are the dimensionalities of the high-dimensional data and the low-dimensional data respectively and K is the number of classes in this dataset. The second part is to compute the cost function Eq. (4) in the objective function which has the complexity $O(N^2D)$. The last part is the logistic regression procedure whose complexity is $O(ND^2)$. Therefore, the total computational complexity of LSPH is: $O(tNMKD + N^2D + tND^2)$, where t is the number of iterations.

3 Experiments

To evaluate our unsupervised LSPH algorithm on the similarity search problem, we perform experiments on two large-scale datasets: **SIFT 1M** with local SIFT descriptors [13] and **GIST 1M** with global GIST descriptors [14]. The SIFT 1M dataset has one million data points, the dimensionality of which is 128. The GIST 1M dataset has the same number of data points, but the dimensionality is 960. All the experiments are performed using Matlab 2013a on a server configured with a 12-core processor and 128GB RAM running the Linux OS.

In our experiments, 10K data points are selected as the queries at random. At the same time, the rest of the dataset can be considered as the image database. During the test phase, if the returned point is in the top 2 percentile points closest to a query, it will be considered as a true neighbor. Since the Hamming distance ranking is fast with hash codes, we will use the Hamming distance ranking to measure our retrieval tasks. The experimental results can be measured by the Mean Average Precision (MAP) and precision-recall curves. Meanwhile, we will also compare the training time and the test time in all the selected algorithms.

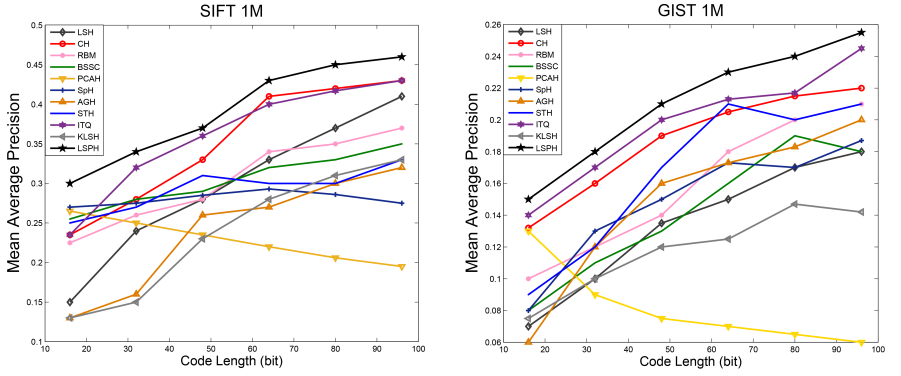

Figure 2: The Mean Average Precision of the compared algorithms on the SIFT 1M and GIST 1M datasets.

3.1 The Selected Methods and Setting

In this part, we compare LSPH with the 10 selected popular hashing methods including LSH [3], BSSC [17], RBM [16], SpH [20], STH [22], AGH [12], ITQ [4], KLSH [7], PCAH [18] and CH [10]. In these methods, for BSSC, through the labeled pairs scheme in the boosting framework, it can obtain weights and thresholds for every hash function. RBM will be trained with several 100-100 hidden layers without fine-tuning. According to KLSH, 500 training samples and the RBF-kernel are used to output the empirical kernel map. In both of them, we always set the scalar parameter σ to an appropriate value on each dataset. In AGH with two-layer, we consider the number of the anchor points k as 200 and the number of the nearest anchors s in sparse coding as 50. CH has the same anchor-based sparse coding setting with AGH. All of the 10 methods are evaluated on different lengths of the codes, *e.g.*, 16, 32, 48, 64, 80 and 96. In the experiments of our LSPH method, we also use the training data as the validation set. Particularly, for each dataset, we select one value from $\{0.01, 0.02, \cdots, 0.10\}$ as the optimal learning rate α through 10-fold cross-validation on the validation set. The regularization parameter λ is selected from $\{10^{-3}, 10^{-2}, \cdots, 10^{2}\}$ via 10-fold cross-validation on the validation set. The regularization parameter δ in the hash function generation is fixed as $\delta = 0.35$ which is chosen from $\{0.05, 0.10, \cdots, 0.50\}$ with the step 0.05.

3.2 Results Comparison

For both of the two datasets, it can be easily seen from Fig. 2 that ITQ always achieves higher Mean Average Precision (MAP) and gets a consistent increasing condition with the change of the code length. MAP of CH is a little lower than ITQ. However, on the **SIFT 1M** dataset, according to SpH and RBM, the accuracy is always "rise-then-fall". BSSC can also obtain competitive accuracies on both datasets. Due to the use of random projection, LSH and KLSH have a low MAP when the code length is short. Moreover, PCAH always gets a decreasing accuracy when the code length increases. For our method LSPH, it achieves the highest performance among all the compared methods on both **SIFT 1M** and **GIST 1M**. The proposed LSPH algorithm can automatically exploit the latent structure of the original data and simultaneously preserve the consistency of distribution between the original data

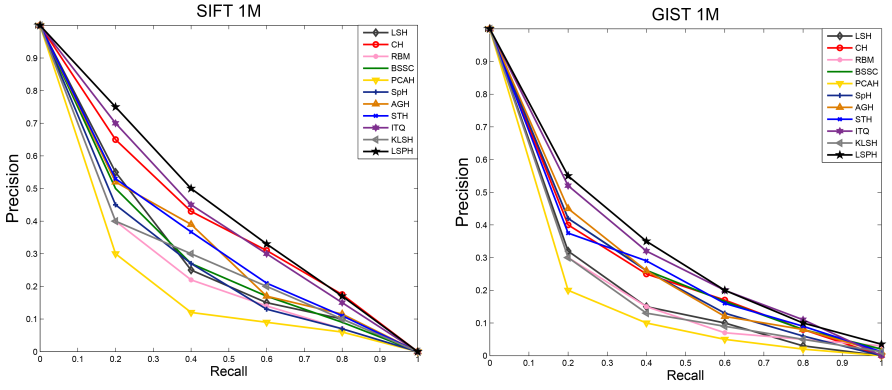

Figure 3: The precision-recall curves of the compared algorithms on the SIFT 1M and GIST 1M datasets for the code of 48 bits.

Table 1: The comparison of MAP, training time and test time of 32 bits and 48 bits of all the compared algorithms on the SIFT 1M dataset.

Methods	**SIFT 1M**					
	32 bits			**48 bits**		
	MAP	**Train time**	**Test Time**	**MAP**	**Train time**	**Test Time**
LSH	0.240	0.3s	1.1μs	0.280	0.6s	1.9μs
KLSH	0.150	10.5s	14.6μs	0.230	10.7s	16.2μs
RBM	0.260	4.5×10^4s	3.3μs	0.280	5.8×10^4s	3.7μs
BSSC	0.280	2.2×10^3s	11.2μs	0.293	2.6×10^3s	13.4μs
PCAH	0.252	6.5s	1.2μs	0.235	7.4s	1.9μs
SpH	0.275	25.8s	28.3μs	0.284	88.2s	101.9μs
AGH	0.161	144.7s	55.7μs	0.267	184.2s	72.0μs
ITQ	0.320	1.4×10^3s	32.1μs	0.360	1.6×10^3s	35.7μs
STH	0.270	1.2×10^3s	17.4μs	0.318	1.8×10^3s	19.8μs
CH	0.280	93.4s	53.5μs	0.330	98.2s	54.4μ
LSPH	**0.340**	1.1×10^3s	20.3μs	**0.376**	1.2×10^3s	22.7μs

and the reduced representations. The above properties of LSPH allow it to achieve better performance in large-scale visual retrieval tasks. In addition, Fig. 3 also illustrates the precision-recall curves of all the methods on both datasets with a code length of 48 bits. It can be obviously observed that our LSPH achieves better performance by comparing the area under the curve (AUC).

Meanwhile, the training and test time for all the methods are listed in Tables 1 and 2. For the training time, LSH and KLSH are the most efficient methods. RBM takes the most time for training, since it is based on a time-consuming deep learning method. Our method LSPH is significantly more efficient than STH, ITQ, BSSC and RBM, but slightly slower than AGH and SpH. Considering the test time, LSH achieves the fastest response as well. AGH and SpH always take more time for the test phase. Our LSPH has the competitive efficiency with STH. Therefore, in general, it can be concluded that LSPH is an effective and relatively efficient method for the large-scale retrieval tasks.

Table 2: The comparison of MAP, training time and test time of 32 bits and 48 bits of all the compared algorithms on the GIST 1M dataset.

Methods	GIST 1M					
	32 bits			48 bits		
	MAP	Train time	Test Time	MAP	Train time	Test Time
LSH	0.107	1.4s	2.7μs	0.135	2.1s	3.0μs
KLSH	0.110	29.5s	27.2μs	0.120	30.7s	38.0μs
RBM	0.123	5.5×10^4s	3.4μs	0.142	6.2×10^4s	3.7μs
BSSC	0.112	3.2×10^3s	13.0μs	0.130	3.8×10^3s	15.1μs
PCAH	0.090	49.2s	2.8μs	0.075	52.3s	3.0μs
SpH	0.130	65.3s	40.2μs	0.148	131.1s	116.3μs
AGH	0.124	242.5s	83.7μs	0.160	279.4s	95.6μs
ITQ	0.170	1.7×10^3s	33.8μs	0.200	1.8×10^3s	36.2μs
STH	0.123	1.9×10^3s	21.3μs	0.171	2.5×10^3s	25.2μ
CH	0.160	194s	64.1μs	0.190	210.5s	71.5μs
LSPH	**0.185**	1.4×10^3s	21.8μs	**0.211**	1.7×10^3s	24.1μs

4 Conclusion

In this paper, we have proposed the Latent Structure Preserving Hashing (LSPH) algorithm, which can find a well-structured low-dimensional data representation through the Nonnegative Matrix Factorization (NMF) with the probabilistic structure preserving regularization part, and then the multi-variable logistic regression is effectively applied to generate the final hash codes. The experimental results on two large-scale datasets demonstrate that our algorithm is a competitive hashing technique for large-scale retrieval tasks.

References

[1] Shumeet Baluja and Michele Covell. Learning to hash: forgiving hash functions and applications. *Data Mining and Knowledge Discovery*, 17(3):402–430, 2008.

[2] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1548–1560, 2011.

[3] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[4] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12): 2916–2929, 2013.

[5] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2957–2964, 2012.

[6] David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression*. 2004.

[7] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *International Conference on Computer Vision*, pages 2130–2137, 2009.

[8] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2000.

[9] Ping Li, Jiajun Bu, Yi Yang, Rongrong Ji, Chun Chen, and Deng Cai. Discriminative orthogonal nonnegative matrix factorization with flexibility for data representation. *Expert Systems with Applications*, 41(4):1283–1293, 2014.

[10] Yue Lin, Rong Jin, Deng Cai, Shuicheng Yan, and Xuelong Li. Compressed hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 446–451, 2013.

[11] Li Liu, Mengyang Yu, and Ling Shao. Multiview alignment hashing for efficient image search. *IEEE Transactions on Image Processing*, 24(3):956–966, 2015.

[12] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *International Conference on Machine Learning*, pages 1–8, 2011.

[13] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[14] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3): 145–175, 2001.

[15] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[16] Ruslan Salakhutdinov and Geoffrey E Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *International Conference on Artificial Intelligence and Statistics*, pages 412–419, 2007.

[17] Gregory Shakhnarovich. *Learning task-specific similarity*. PhD thesis, Massachusetts Institute of Technology, 2005.

[18] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34 (12):2393–2406, 2012.

[19] Qi Wang, Guokang Zhu, and Yuan Yuan. Statistical quantization for similarity search. *Computer Vision and Image Understanding*, 124:22–30, 2014.

[20] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, pages 1753–1760, 2009.

[21] Felix X Yu, Sanjiv Kumar, Yunchao Gong, and Shih-Fu Chang. Circulant binary embedding. *arXiv preprint arXiv:1405.3162*, 2014.

[22] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Self-taught hashing for fast similarity search. In *Conference on Special Interest Group on Information Retrieval*, pages 18–25, 2010.

[23] Wenbin Zheng, Yuntao Qian, and Hong Tang. Dimensionality reduction with category information fusion and non-negative matrix factorization for text categorization. In *Artificial Intelligence and Computational Intelligence*, pages 505–512. 2011.