Unsupervised Nonlinear Hashing Using Nyström Method with Improved Sequential Projection Learning

Wenqiang Yang
School of Information Science and Technology,
Sun Yat-Sen University, China
Email: qwenyang [AT] 163.com

Hong Shen
School of Information Science and Technology,
Sun Yat-Sen University, China
School of Computer Science,University of Adelaide, Australia

Abstract—**Hashing is becoming a popular and effective method for nearest neighbor search in large-scale databases, due to its computational and memory efficient. In this paper, we present an efficient unsupervised nonlinear hashing method to transform high-dimensional data to low-dimensional binary data for fast retrieval. Firstly, we use the Nyström method to transform the feature space into nonlinear kernel feature space, to capture the similarity property of the data. Secondly, all training data are formulated to maximize the entropy over each hash bit, which can be relaxed to maximize the variance on each hash bit. Then, we solve the objective function by an improved sequential projection learning method. In each projection learning iteration, we reduce the HAE (Hamming Accumulated Errors) through some pseudolabel pairs generated from all previous learning projections. During the process, we only need to store the sum of covariance matrix instead of the similarity matrix to save memory storage. We also use a more efficient method to update the covariance matrix after each iteration. We carry out extensive experiments on two benchmarks, and demonstrate that the proposed method achieves better performance than some state-of-the-art hashing approaches.**

Keywords - hashing; similarity search; Nyström method;

I. INTRODUCTION

Nearest neighbor search (similarity search) can be formally described as: given a dataset $X = [x_1, x_2, \ldots, x_n] \in R^{d \times n}$, the objective is to find a set of nearest neighbors $R \subset X$ for a given query q. A naive brute force solution to find the nearest neighbors is to compare every item in the database and find the most similar ones under a predefined similarity metric. However, it is impractical in large databases, because the linear complexity is not scalable. To overcome this difficulty, tree-based methods have been intensively studied for similarity search in past decades. However, performances of tree-based methods, such as KD tree [18], R-tree [3], Cover-Tree [2] are good only for low dimensional data, and drastically degraded to linear search in high dimensions, which is called the curse of dimensionality.

How to efficiently search in large databases is critical for many retrieval applications, such as content-based multimedia retrieval, plagiarism analysis, and collaborative filtering. There are numerous challenges for fast nearest neighbor search in large-scale databases, under the constraints of storage limitation and computational time requirement. Currently, hashing-based methods are becoming more promising for similarity search, due to their fast query speed and low storage cost. The basic idea of hashing-based methods is to design a group of functions that map the high-dimensional data to low-dimensional binary data, and simultaneously preserve their similarities. After that, we can simply return all the objects that are hashed into a ball centered around the query binary code by hash lookup [20], which can be finished in sublinear or even constant time.

From learning paradigm, there are three categories hashing method: unsupervised methods, semi-supervised method, and supervised method. Although supervised and semi-supervised hashing method, such as Iterative Quantization Canonical Correlation Analysis (ITQ-CC) [8], Minimal Loss Hashing (MLH) [16], Two-Step Hashing (TSH) [12], Semantic Hashing using Tags and Topic Modeling (SHTTM) [13], Supervised Hashing With Kernels (KSH) [14], Supervised Hashing with Latent Factor (LFH)[26], Semi-Supervised Hashing (SSH) [20] have higher precision than unsupervised hashing methods in semantic similarity search. But unsupervised hashing method are more common in real situation for there is often no label and similarity information for supervised learning. What's more, unsupervised hashing method, such as LSH (Locality Sensitive Hashing) [5], SH (spectral hashing) [21], Self-Taught Hashing (STH) [24], Density Sensitive Hashing (DSH) [9], Anchor Graph Hashing (AGH) [15], Spherical Hashing(SPH) [1], and Principal Component Analysis Hashing (PCAH) [20] also can achieve good performance in practical situation.

LSH (Locality Sensitive Hashing) [5] is a popular approach for approximate similarity search. The idea is to use a family of hash functions which satisfy the locality-sensitive property: $Pr(h(x_i) = h(x_i)) = sim(x_i, x_i)$ [4]. The intuition is that with multiple hash functions, similar objects have a high probability of being hashed into the same bucket, and

dissimilar objects have a high probability of being hashed into different buckets. Its variants [4][11] have been widely studied. Charikar [4] use a random hyperplane r from zero-mean multivariate Gaussian $N(0, I)$ to multiply each data point, which can keep locality-sensitive property: $sim(x_i, x_j) = x_i \cdot x_i$. Kulis [11] designed $sim(x_i, x_j) = k(x_i, x_i) = \Phi(x_i)^T \Phi(x_i)$ for some embedding function $\Phi(x)$ that only uses the kernel matrix via sampling-based method. LSH uses the hash-table structures to filter most false positives and then only search the candidate objects in the same buckets, which can achieve efficient approximate similarity search. However, in practice LSH may lead to very ineffective hash codes [20][23], since the hash functions using data-independent random projection.

SH (Spectral Hashing) [21] (also referred as spectral embedding) has been shown to be one of the state-of-art approaches for compact binary codes construction. SH define the similarity between each pair of points by Gaussian kernel as $w(i, j) = exp(\frac{\| x_i - x_j \|^2}{\varepsilon^2})$ keeping the property of neighbors in input space as neighbors in Hamming space, and also requires the bits to be uncorrelated and balanced. However, the performance of SH degraded tremendously as the number of bits increases.

Graph based method also plays an important role in machine learning system, widely used in information retrieval and classification problem. It also can be easily used in hash method. The key step of graph based method is to build a neighborhood graph. Tony discussed several typical ways to construct a sparse graph in [19]. But for a large scale application, it is impractical for it cost $O(n^2 d)$ time to construct the graph matrix G, and cost $O(n^2)$ memory to store it. Recently, Anchor Graph Hashing (AGH) [15] appears, which use anchor points [1] to maintain the neighborhood structure to construct big graph. The approximate adjacency graph matrix can be easily constructed by a low-rank matrix $G = KK^T$, just as the kernel method. It can be better approximate the semantic similarities, but not perform very well measured by Euler distance.

PCA (Principal Component Analysis) is a linear data transformation technique to minimize the mean square error (MSE), which plays an important role in dimensional reduction and machine learning. We can quantify the principal components by a certain threshold to achieve binary codes. However, there are three drawbacks of the standard PCA and hence corresponding improvements were proposed: 1) PCA is sensitive to outliers, due to it use l_2 norm, so some robust PCA are derived, such as L_1-norm PCA [10] and R_1-PCA [6]. 2) Errors are accumulated with the real-value principal components converted into binary codes as the code length increasing, which is called HAE (Hamming Accumulated Errors) defined in [23], Wang proposed to correct the errors by sequential projection learning method in [20]. 3) The linear transformation can not capture the non-linear relationships, so nonlinear or kernel method [17] can be used to deal with this problem.

To address the drawbacks 2) and 3) simultaneously, we propose an unsupervised nonlinear hashing with improved sequential projection learning (UNHISPL) method. Most of the previous hashing methods didn't consider the HAE except [20]. The differences between our method and [20] are: First, we use Nyström method to transform the features into nonlinear kernel features, to capture the similarity property of the data. Second, we only store the sum of covariance matrix instead of similarity matrix S and the points in pseudolabel pairs, which is more memory efficient. We also use a more efficient method to update the covariance matrix. Third, we study the influences of similar and dissimilar pseudolabel pairs separately in our models, which are equally treated in [20].

II. Unsupervised Linear Hashing With Sequential Projection Learning

The main purpose of linear hash method is to learn a binary code matrix $Y = [y_1, y_2, ..., y_n] \in \{\pm 1\}^{b \times n}$ from training dataset $X = [x_1, x_2, ..., x_n] \in R^{d \times n}$, and a hash function $W = [w_1, w_2, ..., w_b] \in R^{d \times b}$ which can map each point to its hash code ($y_i = sign(W^T x_i)$). In this paper, a lowercase denote a column vector, a capital denote a matrix or a set, and the detail description of symbols are shown in Table I.

A. Linear Projection

Wang [20] proposed an unsupervised linear hashing to maximize the entropy over each hash bit, which can be relaxed to maximum variance, the formula is written as follows:

$$J_W = \sum_i w_i^T X X^T w_i = tr\{W^T X X^T W\} \quad \textbf{(1)}$$
$$s.t\ W^T W = I$$

X is normalized to have zero mean. The orthogonal constraint $W^T W = I$ to decorrelate the hash bits, the solution can be achieved in a single step through eigenvalue decomposition just as the standard PCA. Actually, the solution is the top b eigenvectors of the covariance matrix $C = XX^T$. Suppose that $\lambda_1 \geq \lambda_2 \geq ..... \geq \lambda_b$ are eigenvalues, and $w_1, w_2, ..., w_b$ are their corresponding eigenvectors. Clearly the bigger eigenvalue is, the more information the eigenvector carries ($\lambda_i = var(w_k^T x)$). However, in most practical data sets, the variance is mainly contained in a few top principal directions, after those directions are removed, the rest of picked eigenvalues are small because of the orthogonal constraint $W^T W = I$. Wang revealed that nonorthogonal constraint has more significant performance in [20].

B. Sequential Projection Learning

To learn a nonorthogonal solution, a sequential projection learning method had been proposed in [20]. The idea is quite intuitive, compute the eigenvector w_k one by one. When

[1] Using k-means clustering to obtain m (m<< n) cluster centers as anchor points

Symbols	Description
X	original data matrix, a column denote a point x_i
Y	binary hash code, a column denote a hashing code y_i corresponding to x_i
Z	nonlinear data matrix, transformed from X, a column denote z_i corresponding to x_i with different dimensionality
W	hash function, each column w_i denote a projection vector
M^k	the set of similar pseudolabel pairs generated in k-th projection learning process
D^k	the set of dissimilar pseudolabel pairs generated in k-th projection learning process
$X_L{}^k$	the points appearance in M^k and D^k, $X_L{}^k \in R^{d \times p}$, p is the number of points, d is dimensionality
$C_k{}^M$	the covariance matrix of the similar pseudolabel pairs in M^k
$C_k{}^D$	the covariance matrix of the dissimilar pseudolabel pairs in D^k

Table 1 : Description of symbols

computed a eigenvector, some pseudolabel pairs are generated, which will be used in the next projections learning to correct the wrong partition on those pairs. The method of generating pseudolabel pairs will be described in the next part. Here M^k, D^k denote the set of similar and dissimilar pseudolabel pairs generated in the k-th projection learning. Suppose $X_L{}^k$ contains all the points in M^k and D^k, p denote the number ofpoints. The similarity matrix $S^k \in \{\pm 1, 0\}^{p \times p}$, and the entity $S_{ij}{}^k$ is assigned to 1 and -1 if the corresponding data pair items (x_i, x_j) belong to M^k and D^k, or else assigned to 0. Then the objective function on all previous pseudolabel pairs at k-th iteration can be represented as:

$$R_{w_k} = \sum_{t=0}^{k-1}\left(\sum_{(x_i,x_j)\in M^t} w_k^T x_i x_j^T w_k - \sum_{(x_i,x_j)\in D^t} w_k^T x_i x_j^T w_k \right)$$

$$= \sum_{t=0}^{k-1} (w_k^T X_L^t S^t X_L^{tT} w_k) \qquad \textbf{(2)}$$

Intuitively, it not only desires similar points to have the same signs but also large projection magnitudes, meanwhile dissimilar points not only with different signs but also as far as possible [20]. The objective function for learning the k-th projection contain the main part (1) and pseudolabel pairs' error correction part (2) as following:

$$J_{w_k} = w_k^T (\eta X X^T + \sum_{t=0}^{k-1} \delta^{k-t} X_L^t S^t X_L^{tT}) w_k \qquad \textbf{(3)}$$

Although all the previous pseudolabel pairs have contribution to current projection learning. However, the contribution is decayed exponentially by the factor δ. Note that after each projection learning, the direction is removed from X to minimize the redundancy in bits.[2] The formula (3) can be easily calculated through eigenvalue decomposition.

C. Generate Pseudolabel Pairs

As we known that, in each iterative projection learning process, using a threshold to partition the real-valued projection component into binary bit, always make wrong partition on some pairs.The error accumulated in each iterative projection, which defined as HAE (Hamming Accumulated Errors) in [23]. Suppose in each iterative projection learning, all points are projected on the direction w_k (the blue horizontal line), as shown in Fig.1. and the solid red vertical line is the partition boundary. We can see that $(r-, r+)$ are probably the similar pairs, $(r-, R-), (r+, R+)$ are probably the dissimilar pairs. Concretely, (x_3, x_4), (x_9, x_{10}) are similar, but have different sign of hash bit; (x_1, x_9),(x_3, x_7) ,(x_4, x_{12}), (x_6, x_{10}) are dissimilar, but have the same sign of hash bit. So we can generate the pseudolabel pairs as follow:

$$M = \{(x_i, x_j) | \ h(x_i) \cdot h(x_j) = -1, | w^T x_i | \le b, | w^T x_j | \le b, d(x_i, x_j) \le \zeta\}.$$
$$D = \{(x_i, x_j) | \ h(x_i) \cdot h(x_j) = 1, | w^T x_i | \le b, | w^T x_j | \ge u, d(x_i, x_j) \ge \varepsilon\}. \qquad \textbf{(4)}$$

Where M denote the set of similar pairs, D denote the set of dissimilar pairs, $h(x) = sign(w^T x)$, $d(x_i, x_j)$ is the Euclidean distance between x_i and x_j. Note that, we only select one pair of (x_i, x_j) and (x_j, x_i) in M or D. We add the distance constraints to make sure that they are near true similar or dissimilar pairs on Euclidean distance.

[2] X can be updated by $X = X - w_k w_k^T X$

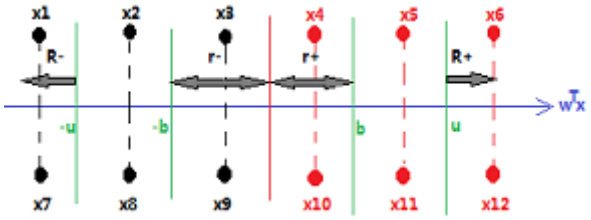


FIGURE 1: GENERATE THE PSEUDOLABEL PAIRS THROUGH THE PRINCIPAL COMPONENT $W^T X$

III. UNSUPERVISED NONLINEAR HASHING WITH IMPROVED SEQUENTIAL PROJECTION LEARNING

D. *Nonlinear Projection*

To address the drawbacks of unsupervised linear hashing, we present an unsupervised nonlinear hashing to effectively capture the latent geometric information of the data. As described in [15][23], the nonlinear hash function can be defined as:

$$h_k(x) = sign(w_k^T z(x)), k = 1, 2, 3, ...b \quad \textbf{(5)}$$

w_k is the k-th projection vector. In [23], $z(x)$ is the x's corresponding column of a regression matrix Z that measures the underlying relationship between the raw samples and the corresponding anchors. In this paper, we use the Nyström method [22][7][25] to compute the nonlinear feature matrix Z which can capture the similarity relationship. The Nyström method is a technique derived from calculating numerical approximations to eigenfunction problems, which use a sampling-based approach to reconstruct the kernel matrix K as:

$$K = E^T A^{-1} E = Z^T Z \quad \textbf{(6)}$$

where $E_{ij} = k(u_i, x_j), E \in R^{m \times n}$, $A_{ij} = k(u_i, u_j), A \in R^{m \times m}$, u_i is a random sample from the dataset as the landmark point. We employ the Gaussian kernel as: $k(x_i, x_j) = exp(-\frac{\| x_i - x_j \|^2}{\varepsilon^2})$, and A is a positive semidefinite matrix, $Z = A^{-1/2} E$. ε is set as in [15].

For a new sample x_n, we can easily get the nonlinear feature $z_n = A^{-\frac{1}{2}} e_n$, $e_n = (k(u_1, x_n), k(u_2, x_n) ..... k(u_m, x_n))^T$. Replace X by Z, the formula (1) can be rewritten as:

$$J_W = \sum_i w_i^T Z Z^T w_i = tr\{W^T Z Z^T W\} \quad \textbf{(7)}$$

E. *Improved Sequential Projection Learning*

Given a set of points $\{z_i\}_{i=1}^n$, we can easily compute the set $(z_i, z_j) \in M$ if z_i and z_j are similar and $(z_i, z_j) \in D$ if dissimilar using formula (4) after each projection learning. From empirical analysis, we expect the distance between each similar pairs being minimized, dissimilar pairs being maximized. We can write the objective function on the pseudolabel pairs at the k-th iteration as:

$$\begin{aligned} min: E\{\| w_k^T z - w_k^T z' \|^2 | M\} &= w_k^T C^M w_k \\ max: E\{\| w_k^T z - w_k^T z' \|^2 | D\} &= w_k^T C^D w_k \end{aligned} \quad \textbf{(8)}$$

where C^M and C^D is the covariance matrix of the similar and dissimilar nonlinear feature differences. Here we do not use the similarity matrix S. This bring several advantages: First, it can save memory, only need $o(d^2)$ storage space. Second, it can deal with more large number of pseudolabels. Third, it benefit for efficient updating operation in the next step.

The finial objective function of our nonlinear hashing method for calculate the k-th projection can be described as follow:

$$J_{w_k} = w_k^T (C_k + \lambda \cdot \sum_{t=0}^{k-1} \delta^{k-t} C_t^D - \mu \cdot \sum_{t=0}^{k-1} \delta^{k-t} C_t^M) w_k \quad \textbf{(9)}$$

where $C_k = Z_k Z_k^T$, and λ, μ is parameters trade-off between "false positive" and "false negative" rates. δ has the same effect as in formula (3). The solution of w_k can be easy computed by eigenvalue decomposition.

To minimize the redundancy in bits, we will subtracting the direction w_k after the k-th projection learning, so the contribution of subspace spanned by that direction is removed from Z . We can update Z as following:

$$Z_{k+1} = Z_k - w_k w_k^T Z_k = U_k Z_k \quad \textbf{(10)}$$

where $U_k = (I - w_k w_k^T)$. We can efficiently update the covariance $C_{k+1} = Z_{k+1} Z_{k+1}^T = (U_k Z_k)(U_k Z_k)^T$, the same updating are applied to C_k^M and C_k^D as follow:

$$\begin{aligned} C_{k+1} &= U_k C_k U_k^T \\ C_{k+1}^M &= U_k C_k^M U_k^T \\ C_{k+1}^D &= U_k C_k^D U_k^T \end{aligned} \quad \textbf{(11)}$$

Note that the complexity of update C_{k+1} is $O(m^3)$, its $O(d^2n)$ in sequential projection learning, $d \approx m \ll n$. The update of C^M, C^D are also more convenient and efficient than the same part in linear hashing with sequential projection learning. The detail pseudo-code is in **Algorithm 1.**

Algorithm 1: Unsupervised Nonlinear Hashing With Improved Sequential Projection Learning Hashing (UNHISPL)

Input: data X , length of hash codes b, parameter λ , μ , δ.

Output: projection matrix W.

Computing the kernel feature Z_1 for all training data using formular (6) , Z_1 are normlized to zero mean;

Initialize $C_1 = Z_1 Z_1^T, C_1^M = 0, C_1^D = 0$;

for k=1 **to** b **do**

Sum the covariance matrix:

$M_k = C_k + \lambda C_k^D - \mu C_k^M$;

Calculate the first eigenvector e of M k and set:

W_k =e;

if k ≥ b break;

Generate the pseudolabel pairs's (satisfying fomulation (4)) difference covariance matrix $\nabla C^M, \nabla C^D$;

Calculate the residual for Z_{k+1} using (10) ;

Update $C_{k+1}^M, C_{k+1}^D, C_{k+1}$ using (11) ;

Sum covariance matrix :

$C_{k+1}^M = \delta \cdot C_{k+1}^M + \nabla C^M$; $C_{k+1}^D = \delta \cdot C_{k+1}^D + \nabla C^D$;

end for

return W.

Remarks: At beginning, there exist no pseudolabel pairs. As k increasing, there are more and more pseudolabel pairs accumulated from all previous projection learning. However, the contribution of pseudolabel pairs is decayed exponentially by the factor δ. Note that W is not orthogonal since the pseudolabel pairs contained the whole space information.

IV. EXPERIMENTS

F. Datasets and Evaluation Metric

At the beginning of this part we will describe two benchmark datasets: SIFT1M and GIST1M [3] . SIFT1M contains one million SIFT features extracted from random images. Each item in the data set is a 128-dimensional vector. For limit of computing resources and convenience of evaluation, we only randomly select 100K samples for training and 1K as query samples. GIST1M contains one million GIST features, each item is a 960-dimensional vector. We also randomly select 100K samples for training and 1K as query samples. We use the same criterion as in [20][9], that a point is considered to be a true neighbor if it lies in the top 1% points closest (measured by the Euclidean distance in the original space). All the experiments are running on my own ordinary PC with Intel Core i7-3770 CPU and 4GB RAM.

Four hashing performance measures are employed in our empirical evaluations: mean average precision (MAP), precision within hamming radius 2 (PH2), Recall and TOP-300. To perform a fair evaluation, we use the same two criteria as [20]. 1. Hamming ranking. All the points in the database are ranked according to their Hamming distance from the query and the desired neighbors are returned from the top of the ranked list. The complexity of hamming ranking is linear. 2. Hash lookup. A lookup table is constructed using the database codes and all the points in the buckets that fall within a small hamming radius *r* of the query are returned, it is very fast when *r* is small.

G. Comparison with Existing Algorithms

Seven state-of-the-art hashing methods for nearest neighbor search are compared. We wrote the code of LSH, used the codes of other methods provided by the respective authors.

1. Locality Sensitive Hashing (LSH [5]): Randomly select projections from a Gaussian distribution with zero-mean and identity covariance, use those projections to construct the hash functions.

2. Principal Component Analysis Hashing (PCAH [20]): Directly use the largest k principal directions of the covariance matrix to construct hash functions.

3. Unsupervised Sequential Projection Learning Hashing (USPLH [20]): Generate pseudolabels at each iteration using a linear sequential projection learning method.

4. Anchor Graph Hashing (AGH [15]): Use a low-rank matrix to approximate the adjacency matrix through anchor graph. AGH with two-layer is used in our comparison for its superior performance over AGH with one-layer.

5. Spectral Hashing (SH [21]): Quantize the values of eigenfunctions computed along PCA directions of the data.

6. Kernelized Locality Sensitive Hashing (KLSH [11]): Generalize the LSH method to the kernel space use sampling-based method.

7. Unsupervised Nonlinear Hashing With Improved Sequential Projection Learning (UNHISPL): Our proposed method in this paper.

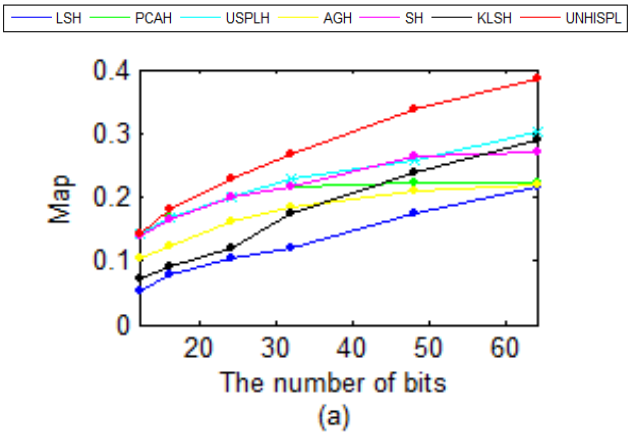
H. Results and Parameters

In the following experiments, we conduct the intensive evaluation on seven hashing methods using two data sets. Fig.2 show the evaluations of MAP, PH2, Recall, Precision-Recall on SIFT1M, and the same evaluations on GIST1M showed in Fig.3. From Fig.2 (a) and Fig.3 (a) , we can see that two random projection based method (LSH,KLSH) have a low MAP when the code length is short. PCAH has little improvement as the code length increase, especially obvious

[3] http://corpus-texmex.irisa.fr/

in Fig.3 (a), this means that the last projections make no use, because of the orthogonal constraint $W^TW = I$, which is consistent with [9]. Our proposed method provides better performance than USPLH show that nonlinear method is better than linear method. Fig.2 (b) and Fig.3 (b) show that all methods exception AGH, PH2 first go up, then go down as the code length increasing. This is main because as the code length increase there are often no nearest neighbor in radius 2 causing precision to zero[4]. But AGH can maintain some nearest neighbors by anchors, its success rate [14] is higher than other methods, so its PH2 increasing as the code length increasing. The Recall curve and Precision-Recall curve are in the case of 32-bits for all method showed in (c) and (d) of Fig.2 and Fig.3, it obvious that our proposed method achieve better performance.

In Table.2 we can see that AGH cost most time in the training process, because it need *k*-means cluster center to be the anchor points. Compare the training time on SIFT1M and GIST1M, We can see that as the dimensionality increase the time of UNHISPL does not change, this is because the nonlinear transformation through Nyström method causing the dimensionality fixed to m [5] on both data sets in our experiments. The query time is accumulated all 1K query points. LSH, PCAH and USPLH are fast than AGH, SH, KLSH and UNHISPL, because those hash methods need some preprocess which cost a little more time.



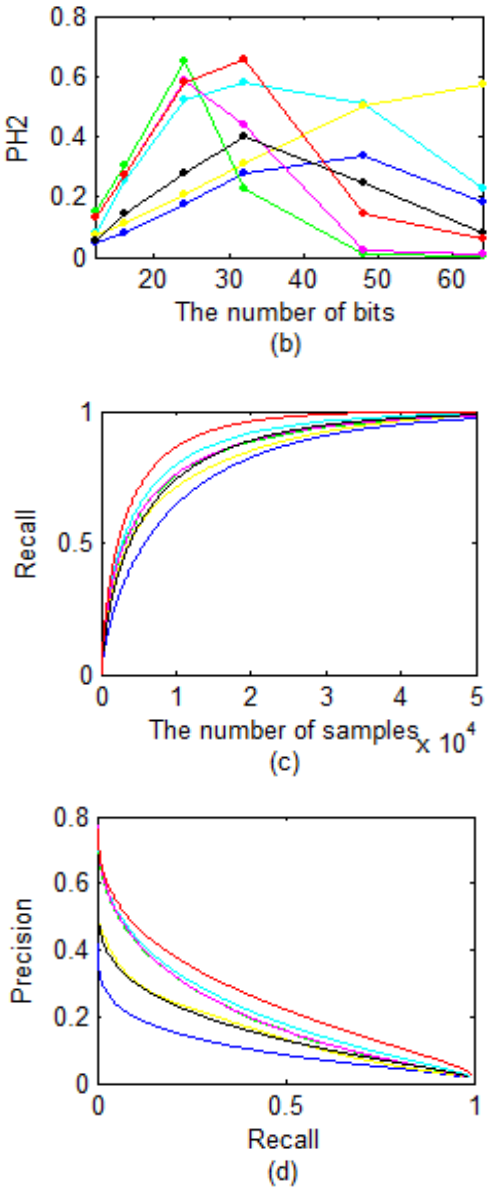
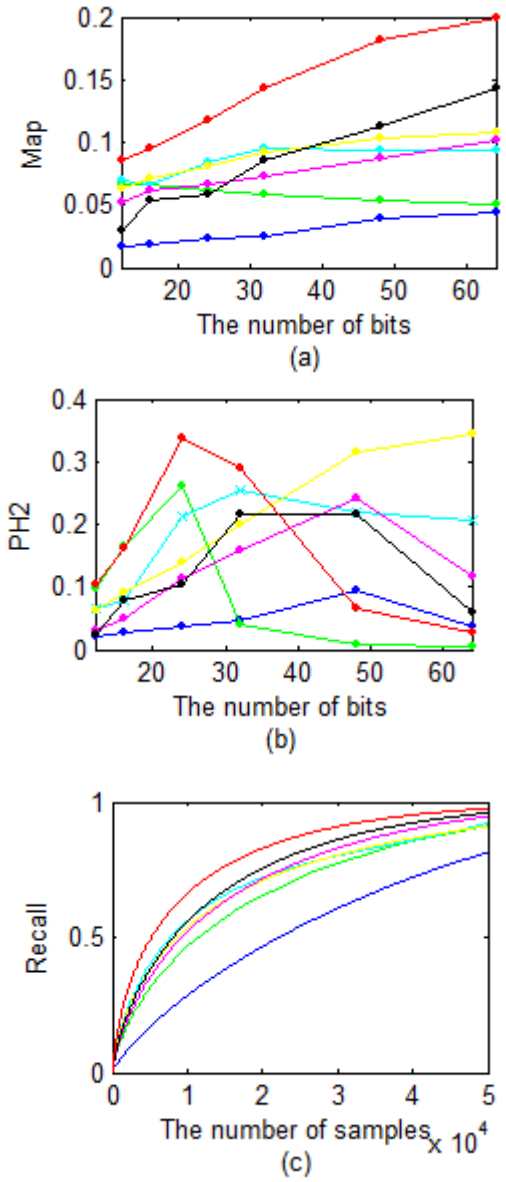


Figure 2: Performance evaluations on SIFT1M:
(a) Map. (b) PH2. (c) Recall. (d) Recall-Precision.

[4] For a query data, if there is no data located in its hamming radius 2 , the precision will be 0.

[5] m is fixed at 300 in our experiments. But for good performance m shold be increasing as the number of training point increasing. From experience, $m \approx \sqrt{n}$ is ok



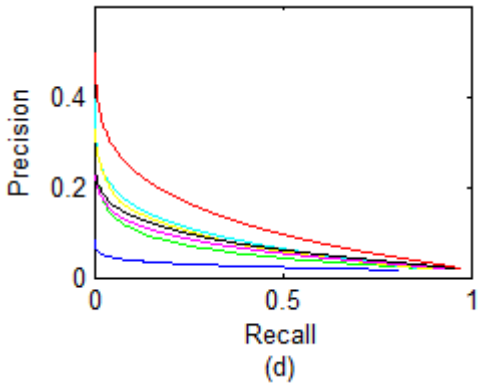


Figure 3: Performance evaluations on GIST1M:
(a) Map. (b) PH2. (c) Recall. (d) Recall-Precision.

Table 2: Time cost on SIFT1M and GIST1M

Method Name	LSH	PCAH	USPLH	AGH	SH	KLSH	UNHISPL
Training Time (s) on SIFT1M	0.1087	0.1408	11.404	129.29	2.4378	1.2081	37.158
Training Time (s) on GIST1M	0.5844	2.0166	185.76	904.57	4.3097	2.8344	37.905
Query Time (s) on SIFT1M	0.0014	0.0013	0.0019	0.0146	0.0223	0.0076	0.0122
Query Time (s) on GIST1M	0.0054	0.0062	0.0093	0.0260	0.0183	0.0186	0.0214

Fig.4 and Fig.5 are the precision of top 300 samples on SIFT1M and GIST1M. We ranked all the point in database according to their Hamming distance from the query point, and select the nearest 300 points to calculate the precisions. With the number of bits increasing, the performance of UNHISPL are more and more good than other methods.

Since UNHISPL use random samples as the landmark points for Nyström method in all of the experiments, the results of UNHISPL are the average of 5 trials. The number of landmark points are fixed to 300 in our experiments. As our experience that using the centers of the k-means clustering as the landmark points can significantly improve the accuracy of the Nyström method. However, it may cost an enormous amount of time to wait for the k-means clustering to convergence, which cannot be tolerated for large databases.

We use cross validation to determine the parameters λ, μ, and set $\lambda = 1.0$, $\mu = 0.5$ and $\delta = 0.9$ in our experiments. We only generate 500 samples from each of the four boundary in each iteration for USPLH and our method. We set the iteration number of k-means cluster at 5 for AGH , and m=300 for AGH and our method. The parameters in other methods are recommended by the corresponding author.

V. CONCLUSIONS

In this paper, we derived an unsupervised nonlinear hashing method for high dimensional nearest neighbor search. The proposed method can capture the non-linear relationships through Nyström method, which can also reduce the dimensionality to the number of landmark points. The proposed improved sequential projection learning method are convenient to update operation. The generated pseudolabel pairs contribute to achieve a good performance by reducing the HAE through correcting some wrong partition pairs. Experiments show that our proposed method performs better than some of the state-of-the-art methods in Euclidean distance for nearest neighbor search.

ACKNOWLEDGMENT

This work is supported by The 985 Project funding of Sun Yat-sen University, Australian Research Council (ARC) Discovery Project DP150104871 and National Science Foundation of China General Projects funding 6117023. The corresponding author is Hong Shen.

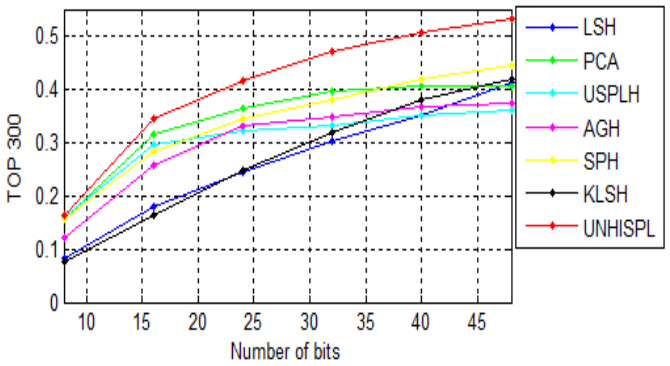


Figure 4: The Precision of Top 300 samples on SIFT1M

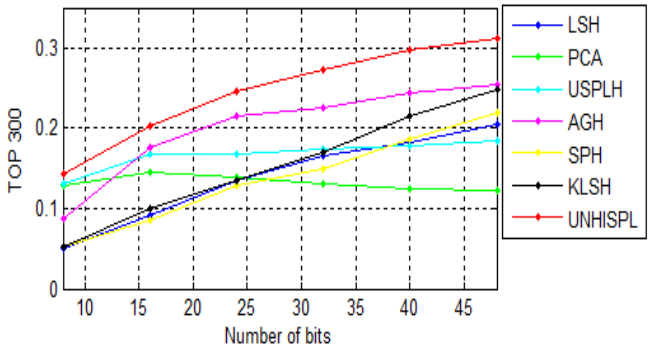


Figure 5: The Precision of Top 300 samples on GIST1M

REFERENCES

[1] Y Heo J P, Lee Y, He J, et al. Spherical hashing[C]//Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012: 2957-2964.

[2] Sham Kakade Alina Beygelzimer. Cover trees for nearest neighbor. Proceedings of the thiry-fourth annual ACM symposium on Theory of computing:97—104, 2006.

[3] Lars Arge, Mark De Berg, Herman Haverkort, Ke Yi. The priority R-tree: A practically efficient and worst-case optimal R-tree. ACM Transactions on Algorithms (TALG), 4(1):9, 2008.

[4] Moses S Charikar. Similarity estimation techniques from rounding algorithms. Proceedings of the thiry-fourth annual ACM symposium on Theory of computing:380—388, 2002.

[5] Mayur Datar, Nicole Immorlica, Piotr Indyk, Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. Proceedings of the twentieth annual symposium on Computational geometry:253—262, 2004.

[6] Chris Ding, Ding Zhou, Xiaofeng He, Hongyuan Zha. R 1-PCA: rotational invariant L 1-norm principal component analysis for robust subspace factorization. Proceedings of the 23rd international conference on Machine learning:281—288, 2006.

[7] Charless Fowlkes, Serge Belongie, Fan Chung, Jitendra Malik. Spectral grouping using the Nystrom method. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26(2):214—225, 2004.

[8] Yunchao Gong, Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on:817—824, 2011.

[9] Zhongming Jin, Cheng Li, Yue Lin, Deng Cai. Density Sensitive Hashing. , 2012.

[10] Qifa Ke, Takeo Kanade. Robust L 1 norm factorization in the presence of outliers and missing data by alternative convex programming. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 1:739—746, 2005.

[11] Brian Kulis, Kristen Grauman. Kernelized locality-sensitive hashing. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 34(6):1092—1104, 2012.

[12] Guosheng Lin, Chunhua Shen, David Suter, Anton van den Hengel. A general two-step approach to learning-based hashing. Computer Vision (ICCV), 2013 IEEE International Conference on:2552—2559, 2013.

[13] Qifan Wang, Dan Zhang, and Luo Si. Semantic hashing using tags and topic modeling. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pages 213–222. ACM, 2013.

[14] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, Shih-Fu Chang. Supervised hashing with kernels. Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on:2074—2081, 2012.

[15] Wei Liu, Jun Wang, Sanjiv Kumar, Shih-Fu Chang. Hashing with graphs. Proceedings of the 28th International Conference on Machine Learning (ICML-11):1—8, 2011.

[16] Mohammad Norouzi, David M Blei. Minimal loss hashing for compact binary codes. Proceedings of the 28th International Conference on Machine Learning (ICML-11):353—360, 2011.

[17] Bernhard Schölkopf , Alexander Smola , Klaus-Robert Müller . Nonlinear component analysis as a kernel eigenvalue problem. Neural computation, 10(5):1299—1319, 1998.

[18] Chanop Silpa-Anan, Richard Hartley. Optimised KD-trees for fast image descriptor matching. Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on:1—8, 2008.

[19] Jun Wang Tony Jebara. Graph Construction and b-Matching for Semi-Supervised Learning. , 2009.

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

[21] Yair Weiss, Antonio Torralba, Rob Fergus. Spectral hashing. Advances in neural information processing systems:1753—1760, 2009.

[22] Christopher Williams, Matthias Seeger. Using the Nyström method to speed up kernel machines. Proceedings of the 14th Annual Conference on Neural Information Processing Systems, (EPFL-CONF-161322):682—688, 2001.

[23] Chenxia Wu, Jianke Zhu, Deng Cai, Chun Chen, Jiajun Bu. Semi-supervised nonlinear hashing using bootstrap sequential projection learning. Knowledge and Data Engineering, IEEE Transactions on, 25(6):1380—1393, 2013.

[24] Dell Zhang, Jun Wang, Deng Cai, Jinsong Lu. Self-taught hashing for fast similarity search. Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval:18—25, 2010.

[25] Kai Zhang, Ivor W Tsang, James T Kwok. Improved Nyström low - rank approximation and error analysis. Proceedings of the 25th international conference on Machine learning:1232—1239, 2008.

[26] Peichao Zhang, Wei Zhang, Wu-Jun Li, Minyi Guo. Supervised Hashing with Latent Factor Models. , 2014.