Semi-randomized Hashing for Large Scale Data Retrieval

Semi-randomized Hashing for Large Scale Data Retrieval

Haichuan Yang*, Xiao Bai*, Jun Zhou†, Peng Ren‡, Jian Cheng§ and Lu Bai¶
*School of Computer Since and Engineering, Beihang University, China
†School of Information and Communication Technology, Griffith University, Australia
‡College of Information and Control Engineering, China University of Petroleum, China
§Institute of Automation, Chinese Academy of Sciences, China
¶Department of Computer Science, University of York, UK

***Abstract*—In information retrieval, efficient accomplishing the nearest neighbor search on large scale database is a great challenge. Hashing based indexing methods represent each data instance as a binary string to retrieve the approximate nearest neighbors. In this paper, we present a semi-randomized hashing approach to preserve the Euclidean distance by binary codes. Euclidean distance preserving is a classic research problem in hashing. Most hashing methods used purely randomized or optimized learning strategy to achieve this goal. Our method, on the other hand, combines both randomized and optimized strategies. It starts from generating multiple random vectors, and then approximates them by a single projection vector. In the quantization step, it uses the orthogonal transformation to minimize an upper bound of the deviation between real-valued vectors and binary codes. The proposed method overcomes the problem that randomized hash functions are isolated from the data distribution. What's more, our method supports an arbitrary number of hash functions, which is beneficial in building better hashing methods. The experiments show that our approach outperforms the alternative state-of-the-art methods for retrieval on the large scale dataset.**

I. Introduction

The thriving Internet has brought many convenience to people, but at the same time, it also improves the difficulty of handling the large scale data. Actually, the image and video data augment every minute because of the facility of smart cell-phone and the prosperous online sharing applications. For example, there are more than 12 Million Web-crawled images annotated with around 17,000 sized WordNet hierarchy in ImageNet[1]. Nearest neighbor (NN) search is a very important and basic approach for many data analysis and processing methods [1], [2], [3]. Representing a query as a high-dimensional vector, NN search aims at finding its most similar samples with a certain metric, $e.g.$, Euclidean distance. However, NN search is time-consuming in dealing with large amount of data, which makes it not feasible in many applications. Therefore, approximate nearest neighbor (ANN) search has become very popular. One of the most common and effective methods for ANN search is hashing. Hashing methods map a high dimensional vector of a query to a binary code string. Then samples with similar hash codes can be retrieved as the approximate nearest neighbors of the query [4]. Such binary codes can be used for indexing, so that data can be retrieved if they are in a small radius of Hamming distance to the query. Directly linear search with binary code is economic in both time and space costs.

For most proposed methods, the strategy used to obtain hash functions can be classified as randomized or optimized. For randomized strategy, the similarity preserving ability of hash code is guaranteed by intrinsic mathematical property. For example, locality sensitive hashing (LSH) generates hash codes whose collision probability equals to the given similarity $S \in [0, 1]$. While similarity measure varies, LSH has been proposed for angular similarity [5], l_p distance [6] and Jaccard coefficient [7]. Another representative hashing method using randomized strategy is shift invariant kernel hashing (SIKH) [8], which uses random Fourier features to make the expected Hamming distance be coincide with the value of shift-invariant kernel such as Gaussian kernel. Variations of LSH also include kernelized locality sensitive hashing (KLSH) [9] and non-metric locality sensitive hashing [10]. Randomized strategy based hashing usually needs relatively long hash codes to attain a satisfactory accuracy. This results in a large and sparse hash table that is inefficient for searching. Optimized strategy defines an objective function and adopts optimization methods to get a solution within a training set. A representative hashing scheme in this category is spectral hashing (SH) [11] which treats the hashing objective as a graph partitioning problem, and solves it by spectral method. Principal component analysis has also been widely used in many hashing methods with optimized strategy. PCA-Direct [12] directly thresholds the results after performing the principal component analysis. PCA-RR [13], PCA-ITQ [12], and isotropic hashing [14] transform the result from PCA with an orthogonal matrix, but with different objectives.

Both randomized strategy and optimized strategy have their advantages and disadvantages. For randomized strategy, since the distribution of data is not taken into consideration, using completely randomized hash functions to preserve the original similarity is not very effective when the code length is short. However, the mathematical property of this strategy guarantees that the extent of similarity preservation will consistently increase when the code length becomes larger [6]. On the contrary, optimized strategy learns hash functions from a training set, and its performance under compact hash code is better than the methods with randomized hash functions. But when the code length is long enough, the performance of optimized strategy based hashing methods nearly have no gain [8].

[1]http://www.image-net.org

In this paper, we propose a novel semi-randomized strategy to learn hash functions. Just like many hashing approaches, our method follows a two-stage strategy [15] to generate binary codes. In the first stage, we adopt an unsupervised objective which uses the Hamming distance to reconstruct the Euclidean distance. Being different from conventional randomized and optimized strategies, our method firstly generates random hyperplanes based on p-stable distribution [16], and then refines them based on a training set by multidimensional scaling (MDS) [17]. On the one hand, we improve the reconstructive strength of the completely random hash functions by considering the data distribution of a training set. On the other hand, we can learn arbitrary number of hash functions so as to construct multiple hash tables [6], [18] and perform bit-selection [19]. In the second stage, we quantize the real values into binary codes. To this end, the conventional $sign$ function is used and an orthogonal matrix is learned to minimize an upper bound of deviation between real-valued vectors and binary codes by extending the algorithm in [12].

In the rest of the paper, we firstly give formally definition of our objective, and then describe the proposed semi-randomized hashing method. Finally, we show experimental results and draw conclusions.

II. Problem Definition

In this section, we introduce the concept of Euclidean distance preserving hashing, and how an objective function of real valued relaxation can be supported by metric MDS and random projections based on p-stable distribution. This forms the theoretical foundation for the proposed semi-randomized hashing approach.

A. Euclidean Distance Preserving Hashing

Given a training set X with n data samples, with each sample being a d-dimensional vector. Represent X as a $d \times n$ matrix, and $Y \in \{-1, 1\}^{r \times n}$ contains the binary codes for the training set. Denote $\{h_k(\cdot)\}_{k=1}^{r}$ as r hash functions, and $h_k : \mathbb{R}^d \rightarrow \{-1, 1\}$. Let x_i be the i-th column of X, and y_i be the i-th column of Y, then the binary code for x_i is $y_i = [h_1(x_i), h_2(x_i), ..., h_r(x_i)]^T$.

Our objective is to generate binary codes Y that preserves the Euclidean distance between samples in X. More specifically, for any data samples x_i and x_j, the squared Euclidean distance[2] between their binary codes $\|y_i - y_j\|^2$ should be related to the squared Euclidean distance between themselves. We formulate this objective as:

$$\min \sum_{i,j}^{n} (\|x_i - x_j\|^2 - s\|y_i - y_j\|^2)^2 \tag{1}$$

where the scale factor s is a positive number for tuning the the disagreement between ranges of $\|x_i - x_j\|^2$ and $\|y_i - y_j\|^2$.

B. Real-valued Relaxation

Solving problem (1) is nontrivial because the binary vectors make the objective function indifferentiable. An intuitive solution is removing the binary constraint and getting an intermediate result $z_i \in \mathbb{R}^r, i = 1, 2, ..., n$, which leads to a new objective function as

$$\min \sum_{i,j}^{n} (\|x_i - x_j\|^2 - \|z_i - z_j\|^2)^2 \tag{2}$$

Note that the scale factor is not used any more because it has been absorbed. In the following, we describe some methods to solve this Euclidean distance preserving problem.

Objective (2) can be optimized by metric MDS [17], which aims at getting low dimensional vectors, such that the dimensionality r is lower than the original dimensionality d. This property is very useful in dimensionality reduction. Metric MDS generates the same result as PCA. This also explains why many PCA based hashing methods [13], [14], [12] achieve good performance in retrieving the Euclidean neighbors. These hashing methods can generate compact binary codes, but at the same time, the number of hash functions is no more than d. It can be seen that using metric MDS (or PCA) is an optimized strategy in hashing.

Metric MDS does not work if hash codes with dimensionality $r > d$ have to be generated. Instead, we can use the randomized strategy based on p-stable distribution. Stable distribution satisfies the following property: if D is a p-stable distribution, $b_1, b_2, ..., b_t$ are t real numbers, and $V_1, V_2, ..., V_t$ are t random variables which are independently and identically drawn from distribution D, then $\sum_i b_i V_i$ will follow the same distribution as $(\sum_i |b_i|^p)^{1/p} V$, where V is a random variable with distribution D and p is a parameter subject to $p \geq 0$ [16]. In [20], the existence of stable distribution is proved when $p \in (0, 2]$. Particularly, the Cauchy distribution and Gaussian distribution are 1-stable distribution and 2-stable distribution respectively. Since the Euclidean distance is closely related to the l_2 norm, our concern is on the 2-stable distribution, $i.e.$, Gaussian distribution.

In order to show the Euclidean distance preserving property of p-stable distribution, we first generate a d-dimensional random vector w whose entries are independently drawn from a standard Gaussian distribution D_s (with zero mean and unit standard deviation). For x_i and x_j defined previously, the distribution of $w^T x_i - w^T x_j = w^T (x_i - x_j)$ follows a Gaussian distribution D_g which has zero mean and variance $\|x_i - x_j\|^2$. Independently generating r different random vectors in this way to compose a $d \times r$ matrix W, we can prove the following proposition.

Proposition 1. *For arbitrary $W^T(x_i - x_j)$, $\frac{1}{r}\|W^T(x_i - x_j)\|^2$ is an estimator of the variance of D_g, i.e., $\|x_i - x_j\|^2$. Furthermore, We can get the expectation and variance of the random variable $\frac{1}{r}\|W^T(x_i - x_j)\|^2$*

$$E(\frac{1}{r}\|W^T(x_i - x_j)\|^2) = \|x_i - x_j\|^2 \tag{3}$$

$$Var(\frac{1}{r}\|W^T(x_i - x_j)\|^2) = \frac{2}{r}\|x_i - x_j\|^4 \tag{4}$$

Proof: According to the property of p-stable distribution, the r entries of the vector $W^T(x_i - x_j)$ are independent of

[2] In some papers, Hamming distance is used for this definition, which serves for the same purpose as the Euclidean distance in our method.

each other and follow the same distribution D_g. Let them be $w_1, w_2, ..., w_r$. Then

$$\begin{aligned} &\mathrm{E}[\frac{1}{r}\|W^T(x_i - x_j)\|^2] = \frac{1}{r}\sum_{k=1}^{r}\mathrm{E}(w_k^T(x_i - x_j))^2 \\ &= \frac{1}{r}\sum_{k=1}^{r}((\mathrm{E}(w_k^T(x_i - x_j)))^2 + \mathrm{Var}(w_k^T(x_i - x_j))) \\ &= \|x_i - x_j\|^2 \end{aligned} \tag{5}$$

To compute the variance, we firstly review the chi-squared (χ^2) distribution which is the sum of t squared independent random variables following D_s. Parameter t is the degrees of freedom, and the variance of chi-squared distribution with t degrees of freedom is $2t$. Since $w_k^T(x_i - x_j)$ follows the Gaussian distribution D_g, transforming them to the random variable following the stardand Gaussian distribution D_s, we have

$$\begin{aligned} &\mathrm{Var}(\frac{1}{r}\|W^T(x_i - x_j)\|^2) \\ &= \frac{\|x_i - x_j\|^4}{r^2}\mathrm{Var}(\sum_{k=1}^{r}(w_k^T(x_i - x_j)/\|x_i - x_j\|)^2) \\ &= \frac{\|x_i - x_j\|^4}{r^2}2r \end{aligned} \tag{6}$$

■

Equation (3) shows that this is an unbiased estimate. Equation (4) shows that if we want to get a precise estimation, $i.e.$ accurately preserving the Euclidean distance, r should be large enough. In p-stable distribution based LSH [6], the random vector w_k is directly used as the projection vector for each hash function, so r is the length of hash code. In addition, equation (4) shows larger r gives smaller variance, so LSH performs better with longer hash codes.

III. Semi-randomized Hashing

In the previous section, we discussed optimized and randomized strategies, $i.e.$, MDS and random projection, for generating r-dimensional real-valued vectors to preserve the Euclidean distance. Both strategies have their shortcomings. Therefore, we present a semi-randomized hashing to make them complementary.

A. Multiple Random Samples Per Bit

As an Euclidean distance preserving method, LSH uses one random vector to project the data to one hash bit, whose ability of Euclidean distance preserving is based on equation (3) and (4). This property makes LSH not perform well with compact hash codes. The primary reason is that it has not considered the data distribution. At the same time, the randomness means there is no constraint on the number of hash functions, which is a favorable property for a lot of applications. On the contrary, using MDS can give a set of optimal projection vectors for the training data, but the amount of these projection vectors can not be larger than d. For dimensionality reduction method, this is not a problem because its objective is to reduce the dimensionality anyway. However, sometimes we do need more than d hash bits for better hashing performance.

Our solution is also based on Proposition 1 to preserve the Euclidean distance by p-stable random vectors. In LSH, the relationship between random vectors and hash functions is one-to-one. Our approach changes this relation to c-to-one, where c is a positive integer. In the following, we show that each of the r vectors is a linear combination of c random vectors. The weights of this linear combination are learned from the training set. Therefore, we name our method semi-randomized hashing.

Let Q be a $d \times c$ matrix whose each column is a random Gaussian vector as how w is defined previously. Our objective is to find a d-dimensional projection vector u to approximate the projection result of Q:

$$\arg\min_u \sum_{i,j}^{n}(\|Q^T x_i - Q^T x_j\|^2 - (u^T x_i - u^T x_j)^2)^2 \tag{7}$$

Notice this objective is defined on $Q^T X$ and $u^T X$ that can be solved by metric MDS. The most effective method for metric MDS is classical scaling [21]. When $c < n$, we can get the result by the covariance matrix $\Sigma = Q^T X H (Q^T X H)^T$. So we get the $1 \times n$ matrix $u^T X$ by the eigenvector of Σ for the largest eigenvalues:

$$u^T X = l^T (Q^T X H) \tag{8}$$

where c-dimensional vector l is the normalized eigenvector of matrix Σ corresponding to the largest eigenvalue. Suppose the data matrix X has zero mean, $i.e.$, $X = XH$. According to equation (8), we get the optimal solution:

$$u = Ql \tag{9}$$

Specially, if the rank of $Q^T X (Q^T X)^T$ is 1, u acts exactly the same as Q.

Repeat the above steps by r times, we can get a $d \times r$ matrix U, with its every column being the optimal u resulted from (9) by using different random matrix Q. For getting the $r \times n$ real-valued result Z, we have

$$Z = \frac{1}{\sqrt{c \times r}} U^T X \tag{10}$$

Concatenate c different random Q as matrix $\hat{Q}$ which has d rows and $c \times r$ columns. Therefore, $\|z_i - z_j\|^2 = \frac{1}{c \times r}\|U^T x_i - U^T x_j\|^2$ is an approximation for $\frac{1}{c \times r}\|\hat{Q}^T x_i - \hat{Q}^T x_j\|^2$, which is the estimator with variance $\frac{2}{c \times r}\|x_i - x_j\|^4$ according to (4). Here r can be any positive integer.

B. Binary Quantization and Multiple Hash Tables

We have already got the semi-randomized Z for the problem in (2). Then we need to transform the real-valued vectors $\{z_i\}_{i=1}^n$ to be binary codes $\{y_i\}_{i=1}^n$. The most common way is $Y = sign(Z)$, where $sign(Z)$ is a matrix with the same size as Z, and $sign(Z)_{ij} = 1$ for $Z_{ij} \geq 0$, and $sign(Z)_{ij} = -1$ otherwise. However, directly using the $sign(\cdot)$ may degrade the performance because we preserve the Euclidean distance $\|x_i - x_j\|$ by $\|z_i - z_j\|$, but the Euclidean distance between binary codes $\|y_i - y_j\|$ may be largely deviated from $\|z_i - z_j\|$. Formally, the deviation δ_{ij} can be defined as

$$\delta_{ij} = |(\|z_i - z_j\| - \sqrt{s}\|y_i - y_j\|)| \tag{11}$$

where s is the scale factor in (1). Based on the triangle inequality of Euclidean distance, it's not hard to find an upper bound of δ_{ij}:

$$\delta_{ij} \leq \|z_i - \sqrt{s}y_i\| + \|z_j - \sqrt{s}y_j\| \tag{12}$$

For arbitrary orthogonal matrix R, Z^T and $Z^T R$ have exactly the same effect because only the Euclidean distance is considered and it is invariant to orthogonal transformation. Since we want to minimize $\|z_i - \sqrt{s}y_i\|$ for $i = 1, 2, ..., n$, using its squared form for convenient computation, we propose a quantization objective:

$$\arg\min_{R} \sum_{i,j}^{n} \|Z^T R - \sqrt{s} \cdot sign(Z^T R)\|_F^2 \tag{13}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. In [12], a similar objective is used but not with the scaling factor. To solve R and s, we iteratively update R using the classic orthogonal procrustes method [22], and update s by setting the partial derivative $\partial Q(\hat{X}R_{(i)})/\partial(\sqrt{s}) = 0$. In each iteration i, we solve the optimal orthogonal matrix $R_{(i)}$ by

$$\arg\min_{R_{(i)}} \|Z^T R_{(i)} - \sqrt{s}_{(i-1)} \cdot sign(Z^T R_{(i-1)})\|_F^2 \tag{14}$$

This can be solved by singular value decomposition (SVD). If the SVD:

$$Z(\sqrt{s}_{(i-1)} sign(Z^T R_{(i-1)})) = F S \bar{F}^T \tag{15}$$

then $R_{(i)}$ should be $F\bar{F}^T$. The optimal $\sqrt{s}_{(i)}$ is

$$\sqrt{s}_{(i)} = \frac{tr(sign(Z^T R_{(i)})^T Z^T R_{(i)})}{tr(sign(Z^T R_{(i)}) sign(Z^T R_{(i)})^T)} \tag{16}$$

We initialize $R_{(0)}$ as a random orthogonal matrix and $s_{(0)}$ as 1. After R and s converge, we can get the binary codes $Y = R^T Z$.

The proposed semi-randomized hashing (SRH) method is summarized in Algorithm 1. Multiple hash tables are usually used in practice to improve the performance [6], [18]. Similar with LSH, our method can construct multiple hash tables by repeating Algorithm 1 by L times where L is the number of hash tables. In this setting, the Hamming distance between binary codes of x_i and x_j is:

$$dist(x_i, x_j) = \min_{t=1,...,L} d_{Hamming}(Y_t(x_i), Y_t(x_j)) \tag{17}$$

Where $Y_t(x_i)$ is the binary code of x_i in the t-th hash table.

Algorithm 1: Semi-randomized hashing

Data: A $d \times n$ data matrix X with zero mean and the length of hashing codes r.
Result: A $r \times n$ binary matrix Y.
for $m = 1$ *to* r **do**
  Independently generate $d \times c$ matrix Q with each column being a Gaussian random vector;
  Set l as the eigenvector of matrix $Q^T X (Q^T X)^T$ corresponding to the largest eigenvalue;
  $u \leftarrow Ql$;
  $U_m \leftarrow u$;
end
$U \leftarrow [U_1, U_2, ..., U_r]$;
Assign $Z = \frac{1}{\sqrt{c \times r}} U^T X$, assign $R_{(0)}$ as random orthogonal matrix, and $\sqrt{s}_{(0)} = 1$;
while R, s *is not converged* **do**
  Update R by equation (14);
  Update $\sqrt{s}$ by equation (16);
end
$Y = sign(R^T Z)$.

IV. Experiments

A. Datasets

We evaluated the performance of our method on several datasets. Their information are introduced as follows.

GIST-1M [23] contains more than 1 million 960-dimensional GIST descriptors [24], including 1,000 query vectors, a learning set with 500,000 instances, and a base set with one million feature vectors. **MNIST** consists of 70,000 handwritten 0∼9 digit images. The images are 28×28 greyscale. The digit in each image is well aligned, so each image can be treated as a 784-dimension feature vector. Since a large portion of images are clean background pixels, each feature vector has a sparse form. **CIFAR-10** [25] consists of 60,000 32×32 color images in 10 classes. The classes include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. We used a 512-dimensional GIST descriptor [24] to represent each image.

B. Evaluation Protocols and Baseline Methods

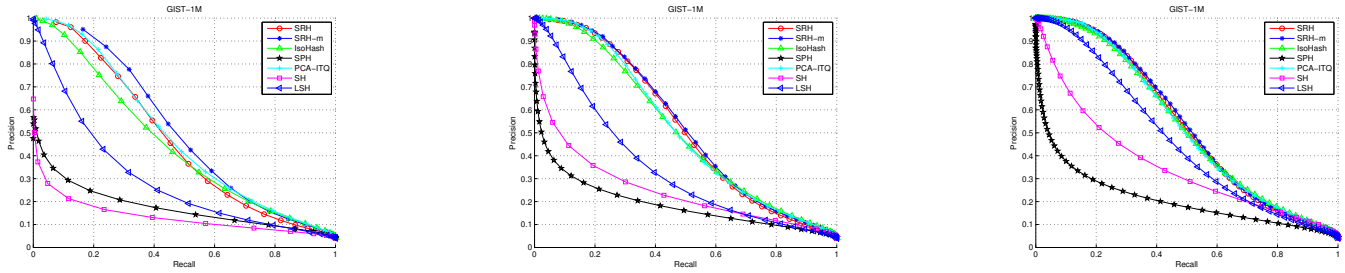
In the experiments, we used the Euclidean neighbors as the ground truth. Similar to [8], we used the average distance of all the query samples to the 50th nearest neighbor as a threshold to determine whether a point in the dataset should be considered as a true positive for a query. For both datasets, we chose 1,000 query samples and 50,000 instances for training hash function. For MNIST and CIFAR-10, all the instances except the query samples are treated as the targets for search. For GIST-1M, the base set is used as the search scope.

We adopted the precision-recall curve and the mean average precision (mAP), $i.e.$, the area under precision-recall curve, to compare the overall performance of different methods. These methods include iterative quantization based on PCA (PCA-ITQ) [12], Isotropic Hashing (IsoHash) [14], spherical hashing (SPH) [26], spectral hashing (SH) [11], and locality sensitive hashing (LSH) [6]. We denote SRH and LSH with multiple hash tables as SRH-m and LSH-m.

There are two free parameters to set in our methods, L which is the number of hash tables and c which let each vector u be a linear combination of c random vectors. In our work, we set $c = 3$ which leads to good result. Although larger c may get better estimation based on equations (4) and (10), the approximation by eigen-decomposition in (8) will be inaccurate. The optimal value of L is 5, which will be analyzed later.

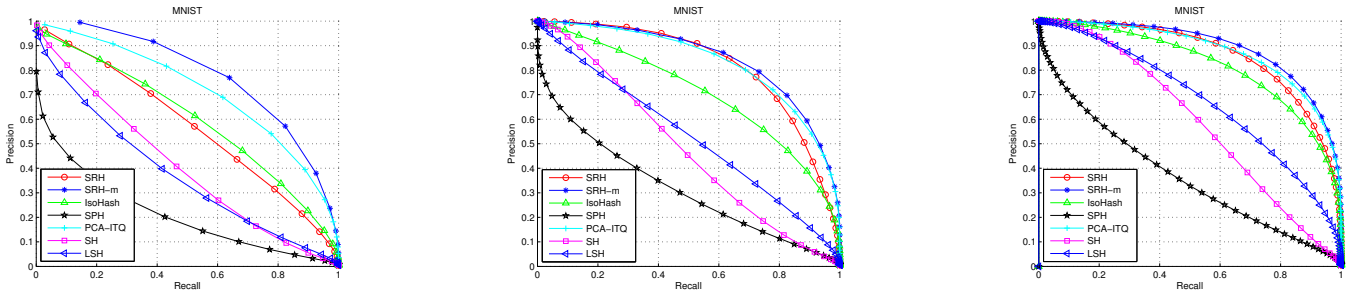
C. Evaluation Results

Figures 1, 2 and 3 show precision-recall curves for Euclidean neighbor retrieval on the three datasets, respectively.



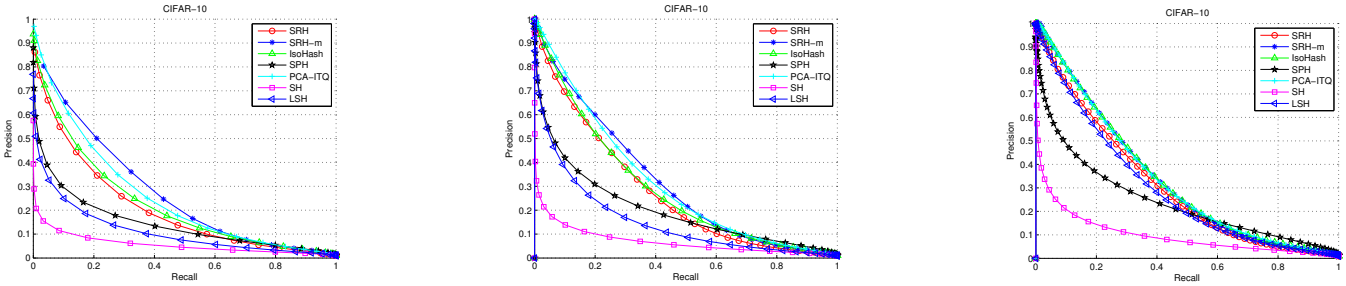
(a) 32 bits (b) 64 bits (c) 128 bits

Fig. 1. Precision-recall curves on GIST-1M, using Euclidean ground truth.



(a) 32 bits (b) 64 bits (c) 128 bits

Fig. 2. Precision-recall curves on MNIST, using Euclidean ground truth.



(a) 32 bits (b) 64 bits (c) 128 bits

Fig. 3. Precision-recall curves on CIFAR-10, using Euclidean ground truth.

On GIST-1M, our method with multiple hash tables SRH-m outperforms all the alternative methods in the case of the code length 32, and our method with single hash table SRH outperforms IsoHash. In the cases of the code length 64 and 128, the precision-recall curves of IsoHash, PCA-ITQ, SRH and SRH-m are very close. For the alternative methods, LSH and SH show significant improvement when the code length increases, while SPH do not work well on GIST-1M. On MNIST, SRH-m also outperforms all the other methods. SRH works better than IsoHash and is close to PCA-ITQ when the code length is larger than 32. On CIFAR-10, the relative performances are similar to the results on GIST-1M, but our method with single hash table SRH outperforms IsoHash.

Lastly, we make comparison of SRH-m and LSH-m, when different numbers of hash tables are used. Table I shows the mAP for LSH-m and SRH-m with the fixed code length $r = 48$ and $c = 3$. It is clear that the proposed method has outperformed LSH-m. We can see that too many hash tables may decrease the mAP because the number of retrieved samples will increase. In most cases, both methods have better performance with more hash tables, and the mAP of SRH-m change slightly when $L \geqq 5$.

D. Computational Cost

Table II shows the training and indexing time on GIST-1M of each method. All experiments were implemented with MATLAB codes, ran on a PC with Core-i7 3.4GHZ CPU and 16GB memory. LSH does not have a training phase because it is a data-independent method. We find that SPH takes the highest training time with $r = 32$ and $r = 64$, though all the methods can complete the training procedure very quickly. Furthermore, the training procedure of our method SRH can be boosted with parallel computing because that training of each hash function is independent to each other. For SRH, LSH,

L	1	3	5	7	9	11
LSH-m	0.34	0.35	0.40	0.40	0.42	0.41
SRH-m	0.49	0.50	0.51	0.51	0.51	0.52

(a) GIST-1M

L	1	3	5	7	9	11
LSH-m	0.50	0.55	0.57	0.64	0.59	0.63
SRH-m	0.74	0.74	0.78	0.78	0.77	0.76

(b) MNIST

L	1	3	5	7	9	11
LSH-m	0.13	0.16	0.19	0.19	0.17	0.18
SRH-m	0.23	0.24	0.26	0.27	0.25	0.25

(a) CIFAR-10

TABLE I. MAP OF LSH-M AND SRH-M WITH PARAMETER L.

PCA-ITQ and IsoHash, the hash function is linear so they have the fastest indexing speed. For SPH and SH, the indexing time are longer, especially for SH which uses a complex nonlinear hash function.

When using multiple hash tables, the training time and indexing time will be L times longer than the single hash table version. This can also be reduced if we generate the hash functions and binary codes in parallel.

Methods	32 bits		64 bits		128 bits	
	Train	Index	Train	Index	Train	Index
SRH	7.31	1.19	16.21	1.96	42.8	3.56
LSH	-	1.18	-	2.05	-	3.73
SH	1.57	13.28	1.95	38.50	2.39	140.05
IsoHash	1.44	1.17	1.51	2.00	2.08	3.58
SPH	12.23	7.05	20.67	8.15	38.95	10.15
PCA-ITQ	2.87	1.12	4.87	1.87	10.14	3.74

TABLE II. TRAINING AND TESTING TIME (SECONDS) ON GIST-1M.

V. CONCLUSION

In this paper, we propose a novel semi-randomized scheme semi-randomized hashing. The principle of our method is based on the property of p-stable distribution which is conventionally used in purely randomized hashing. We refine the generated random projections according to the training data. Semi-randomized hashing shows excellent performance under compact binary codes over randomized hashing such as locality sensitive hashing. Experiments on two public datasets show the superiority of the proposed method compared with the state-of-the-art hashing approaches.

ACKNOWLEDGMENT

This work was supported by the NSFC projects (No. 61370123, 61105002, 61272077 and 61105005), the Australian Research Councils DECRA Projects funding scheme (project ID DE120102948) and Qingdao Fundamental Research Project (No. 13-1-4-256-jch).

REFERENCES

[1] M. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 284–291.

[2] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[3] S. Pandey, A. Broder, F. Chierichetti, V. Josifovski, R. Kumar, and S. Vassilvitskii, "Nearest-neighbor caching for content-match applications," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 441–450.

[4] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the 30th Annual ACM symposium on Theory of computing*, 1998, pp. 604–613.

[5] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 2002, pp. 380–388.

[6] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th Annual Symposium on Computational Geometry*, 2004, pp. 253–262.

[7] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting." in *BMVC*, vol. 810, 2008, pp. 812–815.

[8] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," *The Neural Information Processing Systems*, vol. 22, 2009.

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

[10] Y. Mu and S. Yan, "Non-metric locality-sensitive hashing." in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2010.

[11] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems*, 2008, pp. 1753–1760.

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

[13] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3304–3311.

[14] W. Kong and W.-J. Li, "Isotropic hashing," in *Advances in Neural Information Processing Systems*, 2012, pp. 1655–1663.

[15] ——, "Double-bit quantization for hashing," in *Proceedings of the Twenty-Sixth AAAI Conference*, 2012.

[16] P. Indyk, "Stable distributions, pseudorandom generators, embeddings and data stream computation," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, 2000, pp. 189–197.

[17] T. F. Cox and M. A. Cox, *Multidimensional scaling*. CRC Press, 2010.

[18] X. Liu, J. He, and B. Lang, "Reciprocal hash tables for nearest neighbor search," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[19] X. Liu, J. He, B. Lang, and S.-F. Chang, "Hash bit selection: a unified solution for selection problems in hashing," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2013, pp. 1570–1577.

[20] V. Zolotarev, *One-dimensional stable distributions*. American Mathematical Society, 1986, vol. 65.

[21] G. Young and A. S. Householder, "Discussion of a set of points in terms of their mutual distances," *Psychometrika*, vol. 3, no. 1, pp. 19–22, 1938.

[22] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[23] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 1, pp. 117–128, 2011.

[24] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

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

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