Neurocomputing 275 (2018) 916–923

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Bagging–boosting-based semi-supervised multi-hashing with query-adaptive re-ranking

Wing W.Y. Ng[a], Xiancheng Zhou[a], Xing Tian[a,*], Xizhao Wang[b], Daniel S. Yeung[a]

[a] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[b] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China

ARTICLE INFO

Article history:
Received 19 January 2017
Revised 8 July 2017
Accepted 13 September 2017
Available online 21 September 2017

Communicated by Yongdong Zhang

Keywords:
Semi-supervised information retrieval
Multi-hashing
Bagging
Boosting

ABSTRACT

Hashing-based methods have been widely applied in large scale image retrieval problem due to its high efficiency. In real world applications, it is difficult to require all images in a large database being labeled while unsupervised methods waste information from labeled images. Therefore, semi-supervised hashing methods are proposed to use partially labeled database to train hash functions using both the semantic and the unsupervised information. Multi-hashing methods achieve better precision-recall in comparison to single hashing method. However, current boosting-based multi-hashing methods do not improve performance after a small number of hash tables are created. Therefore, a bagging–boosting-based semi-supervised multi-hashing with query-adaptive re-ranking (BBSHR) is proposed in this paper. In the proposed method, an individual hash table of multi-hashing is trained using the boosting-based BSPLH, such that each hash bit corrects errors made by previous bits. Moreover, we propose a new semi-supervised weighting scheme for the query-adaptive re-ranking. Experimental results show that the proposed method yields better precision and recall rates for given numbers of hash tables and bits.

1. Introduction

The explosive growth of multi-media contents on the Internet creates a huge challenge for image retrieval researches. Image retrieval methods can be categorized into text-based [1–3] and content-based (CBIR) [4–6]. CBIR methods develop rapidly in the past decades. For a large scale CBIR problem, linear search methods may still use too much time and therefore sub-linear methods are needed. Instead of taking a long time to search for exact matches, approximated nearest neighbor search methods [7,8] finding similar images in an approximated manner are much more efficient, especially for very large scale problems and no particular image is targeted. Hashing-based image retrieval methods [9–11] are instances of approximated nearest neighbor search methods which represent images with binary hash codes and have shown to be highly efficient in large scale image searches [12]. For a given query image q, hashing method tries to find its similar by finding images in the database yielding the smallest Hamming distances from q in their hash codes. Therefore, hashing methods generate hash codes for images such that similar images share similar hash codes while dissimilar images have very dissimilar hash codes.

In general, image retrieval performance improves when more hash bits and multiple hash tables are used. However, existing boosting-based multi-hashing methods do not improve and even sometimes reduce retrieval performance after the number of hash tables reaches a certain threshold. Therefore, the bagging–boosting-based semi-supervised multi-hashing method is proposed to address this problem. The proposed method consists of two steps: multi-hashing construction and query-adaptive re-ranking.

Major contributions of this paper include:

- The proposal of a semi-supervised multi-hashing using bagging to relieve the disadvantage of boosting-based multi-hashing methods: new hash table being highly similar to existing one's after a number of tables being created. Then, boosting is used to train individual hash function in each hash table. This hybrid method takes advantages of both bagging and boosting and applies them in different parts of the whole algorithm to maximize their benefits.
- Proposing a semi-supervised weighting scheme for query-adaptive re-ranking to improve retrieval performance of multi-hashing for semi-supervised image retrieval problem.

Related works are introduced in Section 2. The bagging–boosting-based semi-supervised multi-hashing with re-ranking (BBSHR) is proposed in Section 3. Experimental results are shown and discussed in Section 4. Section 5 concludes this paper.

* Corresponding author.
E-mail address: shawntian123@gmail.com (X. Tian).

https://doi.org/10.1016/j.neucom.2017.09.042
0925-2312/

2. Related works

Current hashing methods and multi-hashing methods are introduced in Sections 2.1 and 2.2, respectively.

2.1. Current hashing methods

Hashing methods can be generally divided into three categories: unsupervised, semi-supervised, and supervised hashing methods according to the usage of semantic information. Unsupervised hashing methods [13–17] do not use semantic information from the given database. The Locality Sensitive Hashing (LSH) and its variants [18,19] are the most representative unsupervised hashing methods which create hash functions by random. The Principle component hashing [20] is another unsupervised hashing method which builds hash functions based on the principle component analysis. The iterative quantization hashing (ITQ) [21] finds binary hash codes for images via a minimization of the quantization error between the real-valued data vectors and the binary hash codes. The objective function of the ITQ is minimized by updating the rotation matrix and hash codes iteratively. The Unsupervised Bilinear Local Hashing applies bilinear projections to generate hash codes [16]. Instead of using hash hyperplanes to divide the data space, the spherical hashing finds hyperspheres to partition the image set into different hash buckets (codes) to generate efficient hash codes [17]. Semantic labels in the database provide extra discriminative information and images with the same class label should have the same or very similar hash codes. The LDAHash [22] and the Supervised Hashing with Kernels [23] are instances of supervised hashing methods which require all images in the database to be labeled and use those labels to learn hash functions. The LDAHash applies the linear discriminant analysis to the data features to build hash functions [22]. The Supervised Hashing with Kernels trains hash functions by maximizing Hamming distances between dissimilar data pairs and minimizing Hamming distances between similar data pairs. Supervised hashing methods usually achieve better retrieval performance in comparison to unsupervised hashing methods. Deep learning is also applied to learn effective hashing by the preservation of the semantic information of images [24]. However, in real world applications, image databases are usually partially labeled and requiring all images being labeled is not practical. Moreover, supervised hashing methods tend to overfit when databases cannot provide enough semantic labels.

Therefore, semi-supervised hashing methods [25–28] are proposed to fully utilize the partial labeled images and the other large portion of unlabeled images. The Sequential Projection Learning for Hashing (SPLH) [25] is one of the representative semi-supervised hashing methods which learns hash bits sequentially and corrects error made by the previously learned hash bit. The Bootstrap Sequential Projection Learning Hashing (BSPLH) is another representative semi-supervised hashing method which learns hash bits sequentially and corrects error made by all previous hash bits [26]. The Deep Learning Hashing [29] generates hash codes of training images by the relative similarity graph and learns hash functions from them using the convolution neural network with both visual and semantic information. In [30], two kinds of contextual query expansions (visual world-level and image level) are proposed based on common visual patterns to improve the performance of image retrieval. The topology preserving hashing trains hash functions incorporating the neighborhood ranking information based on data topology [31].

2.2. Current multi-hashing methods

Performances of hashing can be improved by either or both increasing the number of hash bits or/and the number of hash tables. The Complementary Hashing (CH) [32], the Dual Complementary Hashing (DCH) [33], the Boosting Iterative Quantization hashing with query-adaptive re-ranking (BIQH) [34], and the QsRank [35] are instances of multi-hashing methods. Both the CH [32] and the DCH [33] are boosting-based multi-hashing methods. The CH trains hash table to complement error made by the previous hash table while the DCH applies extra boosting during the training of hash bits. The DCH applies the SPLH to train individual hash table. Such that, in addition to complementing error made by the pervious hash bit in the training of the new hash bit in a hash table, the DCH also complements error made by the previous hash table during the training of a new hash table. The BIQH [34] constructs multiple ITQ hash tables by boosting and re-ranks retrieved images using a bit-level weight for each category in the image database. When a query image arrives, the query weight is computed by a weighted (portion of images in its category over all categories) average of category weights of top-N images of retrieved images. The major drawback of the BIQH is the requirement of fully labeled image database which may not be feasible for real-world large scale image retrieval problems. The QsRank [35] constructs undirected graphs using the relationship between the given query and an image in the database. The final retrieval result of the QsRank is constructed using a graph-based ranking method. The Multi-Graph Hashing [36] finds a weight for each graph and the final retrieval result is found by the combination of graphs and their weights.

In summary, current major multi-hashing methods (e.g. the CH, the DCH, and the BIQH) are boosting-based which may not be able to improve retrieval results by increasing the number of hash tables (after a small number). Boosting methods focus on learning of under-learned samples by new hash table and the number of under-learned samples reduces significantly after a small number of iterations. On the other hand, unsupervised hashing methods usually cannot achieve satisfying performance while supervised hashing methods requiring large scale image database to have all images being labeled may be impractical. Therefore, the bagging–boosting-based semi-supervised multi-hashing (BBSHR) method is proposed in this paper to relieve these. The BBSHR increases weights to under-learned samples instead of removing well-learned samples and uses semi-supervised hashing to fully utilize semantic information in partially labeled images.

3. The BBSHR

The Bagging–Boosting-based Semi-supervised Hashing with query-adaptive Re-ranking (BBSHR) consists of three major components: a hybrid semi-supervised multi-hashing to train multiple hash tables, semi-supervised category-specific weight generation, and a semi-supervised query-adaptive re-ranking to order the retrieved images for a given query. These three components will be proposed in Sections 3.1–3.3, respectively.

3.1. Hybrid semi-supervised multi-hashing for hashing tables construction

The proposed hybrid method uses a bagging approach to create multiple databases for training multiple hash tables (Section 3.1.1) and a boosting approach to train hash functions in each hash table (Section 3.1.2). In this way, we relieve the problem of severely reduced number of training samples of boosting-based multi-hashing methods after a small amount of training iterations.

3.1.1. Bagging-based semi-supervised multi-hashing

Bagging is widely used in machine learning for both classification and regression [37,38]. In general, the bagging method creates m training databases by randomly drawing pn images for m times from the unlabeled part of the original database X with n

images [39], where m, n, and $0 < p < 1$ denote the number of bagging databases, the total number of images in X, and the ratio of images being selected for each training database, respectively. Let X_l and X_u be the labeled image set and the unlabeled image set in X, i.e. $X = X_l \cup X_u$ and $X_l \cap X_u = \emptyset$.

Owing to the semi-supervised nature of the semantic image retrieval problems, applying the standard bagging method on Xs may not be efficient. Therefore, we use a semi-supervised bagging method dedicated for our semi-supervised learning environments. Owing to the nature of only a small portion of labeled images available in the database, all images in X_l are added to every training database. Then, pn' images are randomly drew from X_u with replacement to form the tth unlabeled image database $X_{u,\,t}$, where n' denotes the number of unlabeled images X. Then, the tth training database is created by $X_t = X_l \cup X_{u,t}$. These processes are repeated m times to create m individual training databases.

3.1.2. Boosting-based hash functions training per hash table

Each training database is used to train a hash table with K hash functions. The tth hash table (H_t) consists of K hash functions, i.e. $H_t = \{h_{t,1}, h_{t,2}, \ldots, h_{t,K}\}$. The kth hash function of the tth hash table is represented by:

$$h_{t,k}(x_i) = sgn(W_{tk}^T x_i) \tag{1}$$

where T, W_{tk} and $sgn(*)$ denote the transpose of matrix, the projection vector and the sign function which returns -1 when $* < 0$ and $+1$ otherwise. Therefore, the learning of a hash function is to find the projection vector W_{tk} of it.

Hash functions are expected to generate hash codes such that similar images share the same or similar hash codes while dissimilar images yield different hash codes. In other words, the objective of hash function learning is to find projection vectors to minimize the following optimization problem:

$$\min E\{d_t(x_i, x_j)|P\} - E\{d_t(x_i, x_j)|N\} \tag{2}$$

where P and N denote the set of all similar image pairs (i.e. x_i and x_j) and the set of all dissimilar image pairs, respectively. Images of a similar image pair have the same semantic labels while images of a dissimilar image pair have different labels. $E\{*\}$ and $d_t(x_i, x_j)$ denote the average value of all $*$ and the Hamming distance between hash codes x_i and x_j computed using the tth hash table, respectively.

Let S be the semantic similarity matrix of labeled images where S_{ij}=1 if $(x_i, x_j) \in P$ and $S_{ij} = -1$ if $(x_i, x_j) \in N$. Problem (2) is equivalent to the following optimization problem:

$$\min \sum_k \sum_{ij} S_{ij} ||h_{t,k}(x_i) - h_{t,k}(x_j)||^2 \tag{3}$$

Then, Problem (3) is converted into the following maximization problem:

$$\max \sum_k \sum_{ij} S_{ij} h_{t,k}(x_i) h_{t,k}(x_j) \tag{4}$$

By relaxing the sign function condition to the real value output from the hash function (same as [25]), Problem (4) is transformed into the following form:

$$J(W) = \max tr(W_t^T X_l S X_l^T W_t) \tag{5}$$

where $tr()$ denotes the trace of a matrix. Then, similar to the BSPLH, a regularization term $(W_t^T X_t X_t^T W_t)$ with a regularization parameter λ based on the unlabeled images is added to avoid overfitting. Therefore, the final objective function for training a hash table using the tth training database is as follows:

$$J(W) = \max tr(W_t^T X_l S X_l^T W_t + \lambda W_t^T X_t X_t^T W_t) \tag{6}$$

For the bagging-based multi-hashing, we train m hash tables using m training databases. Therefore, the overall optimization problem is as follows:

$$J(W) = \max \sum_t tr(W_t^T X_l S X_l^T W_t + \lambda W_t^T X_t X_t^T W_t) \tag{7}$$

Problem (7) is then rewritten as follows:

$$J(W) = \max \sum_t tr(W_t^T M_t W_t) \tag{8}$$

where $M_t = X_l S X_l^T + X_t X_t^T$. Optimization Problem (8) for training an individual hash table can be solved using the BSPLH [26] procedures as described in Algorithm 1. Algorithm 1 shows the overall bagging and the BSPLH for training m hash tables for the semi-supervised bagging-based multi-hashing construction.

Algorithm 1 Bagging–Boosting-based Semi-supervised Hashing (BBSH).

Input: data X, labeled data X_l, label L for X_l, length of hash code K, number of hash tables m, constants $\alpha, \beta, \lambda, p$
Output: projection matrixes W_t where $t = 1, 2, \ldots m$

```
Get the semantic matrix S using L
for t =1 to m do
    Construct X_t by the semi-supervised bagging method
    S_1 = S, X_{t,1} = X_t
    for k = 1 to K do
        M_t = X_l S_k X_l + λ X_{t,k} X_{t,k}^T
        Extract the first Eigen vector of M_t : w_{t,k}
        Compute the update weight matrix ΔS^k using Eq. (9)
        S_{k+1} = S_1 + ΔS^k
        X_{t,k+1} = X_{t,k} − w_k w_k^T X_{t,k}
        W_{t,k} = w_{t,k}
    end for
end for
```

The element of update weight matrix $\Delta S^k (\Delta S_{ij}^k)$ is computed as follows:

$$\Delta S_{ij}^k = \begin{cases} \dot{A}_{ij} & A_{ij} > 0, S_{ij}^k > 0 \\ B_{ij} & B_{ij} < 0, S_{ij}^k < 0 \\ 0 & otherwise \end{cases} \tag{9}$$

where $A_{ij} = (\alpha k - D_{ij}^k)/2k$, $B_{ij} = (\beta k - D_{ij}^k)/2k$ and $D^k = \sum_{s=1}^k sgn(X_l^T w_s w_s^T X_l)$.

3.2. Semi-supervised category-specific weight generation

The semantic information provided by the semi-supervised training data can further improve the retrieval performance of hashing [40]. However, the re-ranking method in [34] is fully supervised which is not applicable to semi-supervised hashing. Therefore, we propose a new semi-supervised re-ranking based on [34] in this section.

3.2.1. Pseudo-label assignment

The major problem of semi-supervised data is that a large portion of images are unlabeled. Therefore, we propose to assign pseudo-labels to those unlabeled images. For every unlabeled image, its pseudo-label is assigned to be the majority class label of 20% of all nearest labeled images of it. Euclidean distance is used to improve accuracy of nearest images and it is acceptable because pseudo-labels are assigned offline before any training or query.

3.2.2. Category-specific weight computation

For semi-supervised retrieval problems, labeled images sharing the same category label are regarded as similar. For a hash function, images from the same category (without regarding real or pseudo-label) are expected to have the same hash code. In [34], the performance of a hash function classifying images in the same category to the same side is used as an evaluation criterion.

For all images in the category C, the performance of a hash function $h()$ can be evaluated by the category weight (v_C) computed as follows:

$$v_C = \frac{\max(c_-, c_+)}{c_- + c_+} \tag{10}$$

where c_- and c_+ denote the number of images in C yielding $h() = -1$ and the number of images in C yielding $h() = +1$, respectively. v_C is maximum when all images are hashed to one side of the hash function (i.e. all $+1 or -1$) and minimum when $c_- = c_+$. Finally, v_C is adjusted to the range [0, 1] as follows:

$$v_C = (v_C - 0.5) * 2 \tag{11}$$

For each hash table H_t, a performance weight matrix V_t is computed, where $V_t(i, k)$ denotes the category weight of the ith category for the hash function $h_{t,k}()$ computed by Eq. (11). Then, $V_t(i, :)$ denotes the weight vector for the tth hash table of the ith category.

3.3. Semi-supervised query-adaptive re-ranking

Let R_t be the set of images returned by the tth hash table for the query q with Hamming distance less than or equal to a preselected threshold without re-ranking. Let l_i be the label (without regarding real or pseudo-label) of the image x_i in R_t. Then, the weight vector of the tth hash table for the given q ($Z_t \in R^{1\times K}$) is calculated as follows:

$$Z_t = \frac{\sum_{x_i \in R_t} V_t(l_i, :)}{n_t} \tag{12}$$

where n_t denotes the number of images in R_t.

Then, weighted Hamming distances of images are computed by the following equation:

$$d_w(x_i, q) = \sum_{t=1}^{m} \sum_{k=1}^{K} Z_t(k)(h_{t,k}(x_i) - h_{t,k}(q)) \tag{13}$$

where $Z_t(k)$ denotes the kth element of the Z_t vector. Finally, images yielding smallest weighted Hamming distances (d_w) are returned to user.

4. Experiments

In this section, we compare the BBSHR with state-of-the-art hashing methods using three databases: the MNIST, the USPS, and the CIFAR10. Methods in comparisons include: the LSH [18,19], the CH [32], the DCH [33], the BIQH [34], the BSPLH [26], and the SPLH [25].

The USPS and the MNIST are handwritten digits databases consisting of 10 categories: 0–9 digits. The MNIST consists of 70K 28×28-pixel images being represented by 784-dimensional feature vectors. The USPS consists of 9298 16×16-pixels images being represented by 256-dimensional feature vector. The CIFAR10 consists of 60K images distributed in 10 categories. All databases are divided into two parts: 1000 samples as the testing set and the remaining samples as the training set.

The LSH and the CH are representative unsupervised hashing methods which serve as baselines of comparisons. The BIQH is a fully supervised hashing method with a fully supervised query-adaptive re-ranking. All other methods are semi-supervised hashing methods. The BSPLH and the SPLH are representative semi-supervised single table methods while the DCH is a representative boosting-based semi-supervised multi-hashing method. In addition, to emphasize the benefit of the proposed semi-supervised query-adaptive re-ranking, the BBSH is compared. The BBSH is a simplified version of the BBSHR without the semi-supervised query-adaptive re-ranking. For semi-supervised methods, we random select 1000 samples from the training set to form the labeled sample set and the remaining training samples are used to form the unlabeled sample set. In this paper, experiments are repeated for 10 times and their average performances are reported as the final results. Similar to other semantic hashing works, two samples are similar if they share the same label. The flow chat of the BBSHR is shown in Fig. 1.

4.1. Experimental results

In our experiments, recall-precision curves are used to evaluate performances of hashing methods. Figs. 2–6 show recall-precision curves of different methods on the three databases using hash code lengths of 16-bit, 24-bit, 32-bit, 48-bit, and 64-bit, respectively. In addition, the Area Under Curve (AUC) [41] of recall-precision curves are used to provide further numerical comparison among different hashing methods and their statistical significances are also tested by using the t-test.

Average and standard deviation values of AUC of different methods on different databases using different number of hash bits are shown in Table 1. The first row of Table 1 shows the name of the database and the number of bits being used per hash table

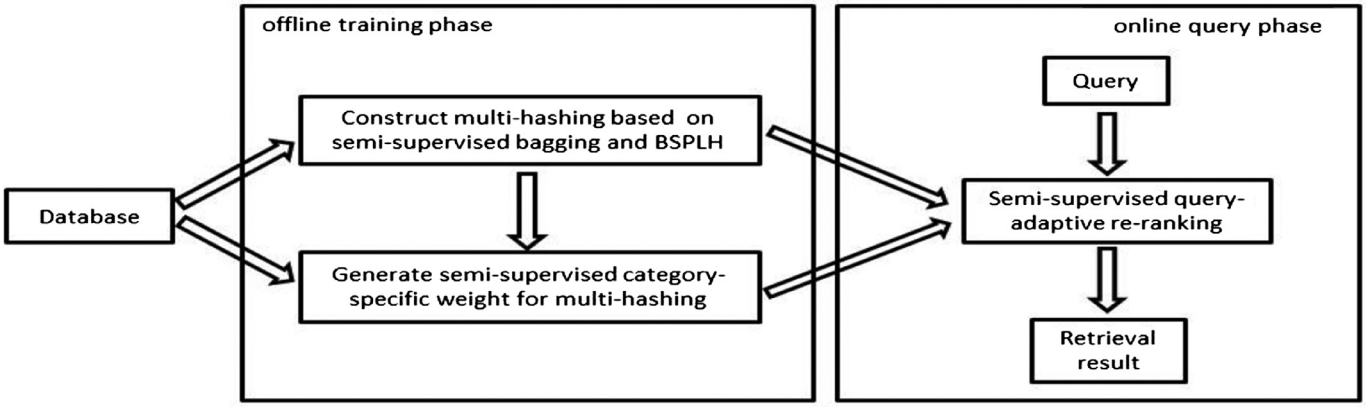

Fig. 1. Flow chart of BBSHR.

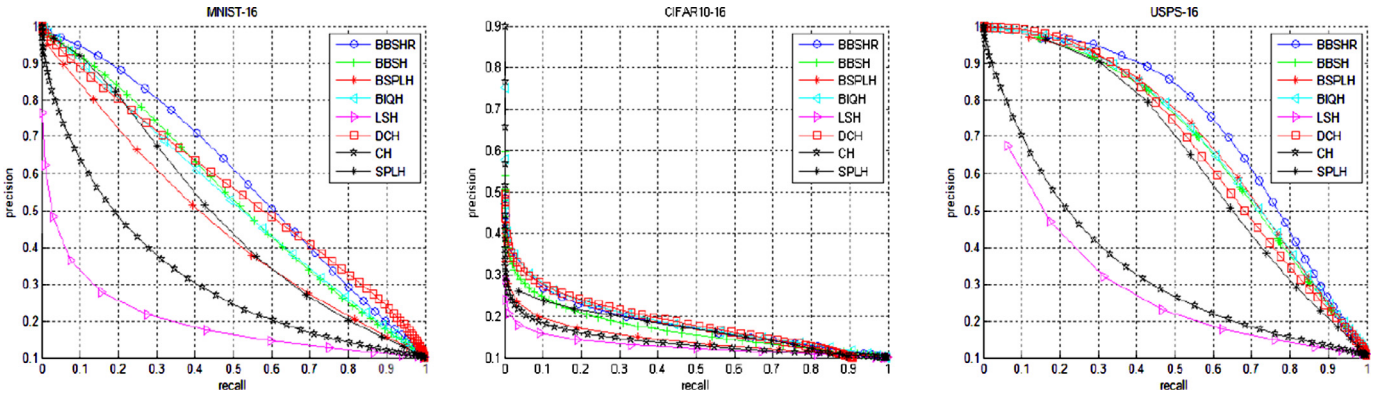

Fig. 2. Recall–precision curves with 16-bit on MNIST (a), CIFAR10 (b), and USPS (c).

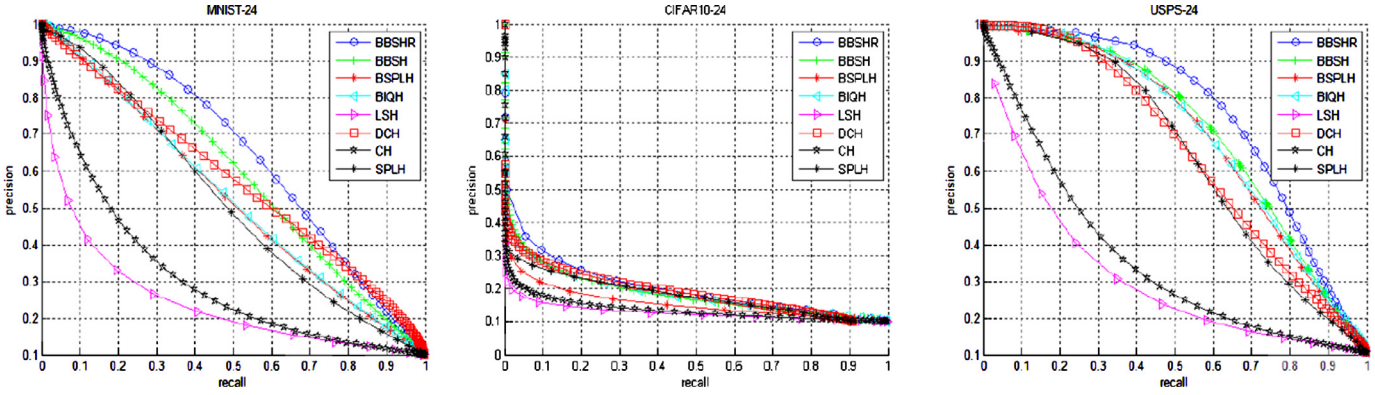

Fig. 3. Recall–precision curves with 24-bit on MNIST (a), CIFAR10 (b), and USPS (c).

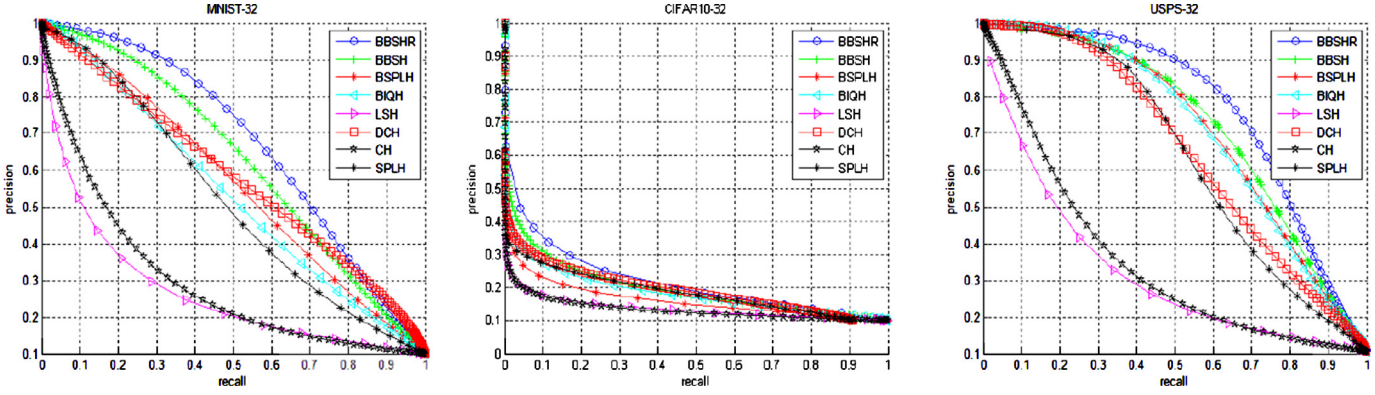

Fig. 4. Recall–precision curves with 32-bit on MNIST (a), CIFAR10 (b), and USPS (c).

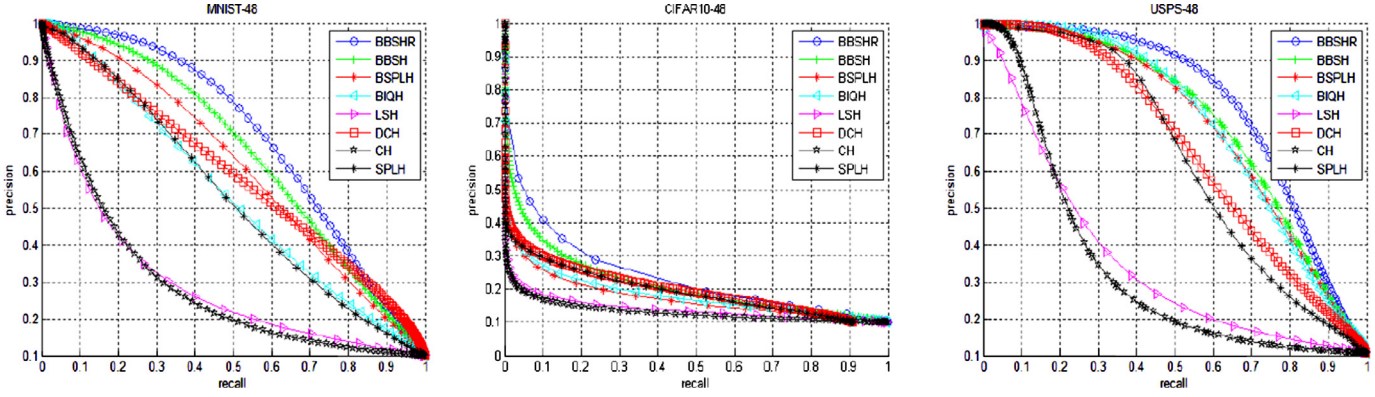

Fig. 5. Recall–precision curves with 48-bit on MNIST (a), CIFAR10 (b), and USPS (c).

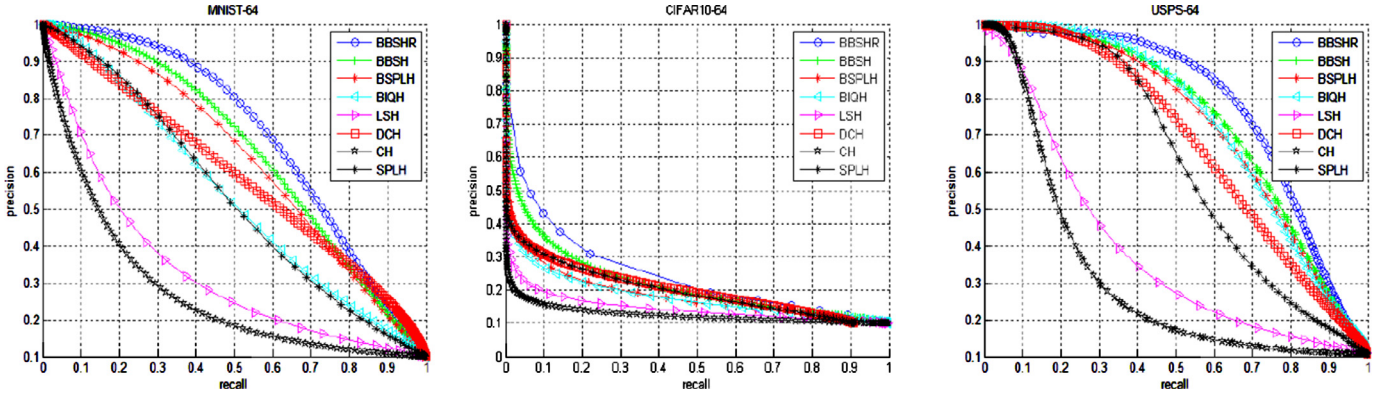

Fig. 6. Recall–precision curves with 64-bit on MNIST (a), CIFAR10 (b), and USPS (c).

Table 1
AUC on MNIST, CIFAR10 and USPS with different hash code length.

	BBSHR	BBSH	BIQH	DCH	CH	BSPLH	SPLH	LSH
MNIST-16	**0.567 ± 0.022**	0.526 ± 0.025*	0.503 ± 0.007*	0.549 ± 0.016&	0.304 ± 0.006*	0.412 ± 0.011*	0.441 ± 0.011*	0.180 ± 0.012*
MNIST-24	**0.634 ± 0.016**	0.590 ± 0.016*	0.509 ± 0.005*	0.567 ± 0.013*	0.298 ± 0.006*	0.489 ± 0.011*	0.483 ± 0.006*	0.224 ± 0.015*
MNIST-32	**0.661 ± 0.018**	0.617 ± 0.020*	0.517 ± 0.008*	0.575 ± 0.012*	0.288 ± 0.003*	0.534 ± 0.011*	0.492 ± 0.008*	0.244 ± 0.013*
MNIST-48	**0.684 ± 0.019**	0.643 ± 0.022*	0.521 ± 0.006*	0.584 ± 0.011*	0.282 ± 0.003*	0.586 ± 0.010*	0.514 ± 0.008*	0.277 ± 0.015*
MNIST-64	**0.695 ± 0.021**	0.659 ± 0.021#	0.525 ± 0.005*	0.588 ± 0.018*	0.270 ± 0.003*	0.620 ± 0.015*	0.522 ± 0.016*	0.315 ± 0.011*
CIFAR10-16	0.183 ± 0.005	0.169 ± 0.005*	**0.188 ± 0.003$**	0.178 ± 0.007#	0.139 ± 0.001*	0.146 ± 0.003*	0.160 ± 0.007*	0.128 ± 0.003*
CIFAR10-24	**0.203 ± 0.006**	0.185 ± 0.005*	0.186 ± 0.002*	0.181 ± 0.008*	0.136 ± 0.001*	0.155 ± 0.003*	0.170 ± 0.006*	0.128 ± 0.005*
CIFAR10-32	**0.218 ± 0.006**	0.197 ± 0.005*	0.185 ± 0.003*	0.184 ± 0.003*	0.133 ± 0.001*	0.162 ± 0.003*	0.176 ± 0.006*	0.137 ± 0.005*
CIFAR10-48	**0.240 ± 0.050**	0.213 ± 0.005*	0.183 ± 0.003*	0.190 ± 0.004*	0.132 ± 0.001*	0.174 ± 0.004*	0.183 ± 0.006*	0.140 ± 0.003*
CIFAR10-64	**0.248 ± 0.005**	0.219 ± 0.004*	0.181 ± 0.002*	0.194 ± 0.006*	0.126 ± 0.001*	0.185 ± 0.004*	0.188 ± 0.007*	0.146 ± 0.005*
USPS-16	**0.709 ± 0.018**	0.640 ± 0.018*	0.667 ± 0.011*	0.657 ± 0.040*	0.349 ± 0.006*	0.644 ± 0.013*	0.601 ± 0.025*	0.252 ± 0.038*
USPS-24	**0.744 ± 0.018**	0.686 ± 0.017*	0.687 ± 0.010*	0.644 ± 0.040*	0.323 ± 0.005*	0.667 ± 0.015*	0.616 ± 0.020*	0.275 ± 0.040*
USPS-32	**0.760 ± 0.020**	0.706 ± 0.023*	0.693 ± 0.013*	0.646 ± 0.032*	0.300 ± 0.002*	0.687 ± 0.009*	0.618 ± 0.022*	0.290 ± 0.040*
USPS-48	**0.772 ± 0.018**	0.722 ± 0.022*	0.715 ± 0.007*	0.650 ± 0.041*	0.269 ± 0.002*	0.704 ± 0.009*	0.620 ± 0.020*	0.325 ± 0.030*
USPS-64	**0.774 ± 0.017**	0.730 ± 0.017*	0.722 ± 0.009*	0.672 ± 0.040*	0.250 ± 0.002*	0.711 ± 0.009*	0.611 ± 0.021*	0.362 ± 0.033*

in a combined form, e.g. MNIST-16 denotes the MNIST database using 16-bit hash codes. The *t*-test is performed for the BBSHR with respect to each of other method. In Table 1, '*', '#', '&' and '$' denote the BBSHR outperforms a particular method with statistical significance of 99.9%, 99%, 95% and less than 50%, respectively. Table 1 shows that the BBSHR outperforms all existing methods in experiments except the BIQH in the CIFAR10-16 experiment with less than 50% statistical significance. Overall, the proposed BBSHR is significantly better than state-of-the-art hashing methods in comparisons.

Both unsupervised methods, i.e. the LSH and the CH, perform the worst among all methods in comparisons. The CH performs slightly better than the LSH because the CH is a multi-hashing method which uses more bits in total. Single table-based methods, i.e. the SPLH and the BSPLH, perform worse than multi-hashing methods except the unsupervised CH. The major reason is that multi-hashing methods use more hash bits in comparison to single table-based methods. It shows the benefits of multi-hashing. The DCH outperforms the CH in all experiments because the DCH is a semi-supervised hashing and uses a boosting-based method for individual hash table training. Although the BIQH uses fully labeled training database, it does not outperform semi-supervised multi-hashing DCH without query-adaptive re-ranking in 7 out of 15 experiments. This shows the major deficiency of the BIQH is the use of unsupervised hashing method in combination of a fully supervised re-ranking. The unsupervised hashing method wastes labeled information while the supervised re-ranking imposes a strong constraint to the BIQH by forcing it to use a fully labeled training database.

The BBSH outperforms the DCH and the BIQH in 12 and 11, respectively, out of all 15 experiments. This shows that both the bagging–boosting-based multi-hashing methods with and without re-ranking, i.e. the BBSHR and the BBSH respectively, are effective. However, without the re-ranking, it is difficult to outperform the BIQH using a fully supervised re-ranking. In contrast, the re-ranking of the BBSHR is designed for semi-supervised databases and more practical to real-world large scale problems. Overall, the BBSHR outperforms the BBSH by 3.93% in average of all experiments.

Another observation is that better performances can be achieved for all hashing methods except the DCH and the CH in the same database when more hash bits are used. This may be caused by the nature of boosting in the DCH and the CH which make them cannot improve after a number of hash tables being created owing to the out of useful training samples issue. This is particularly significant when more bits per table is used because the first few hash tables learn well using more bits and a lot of training samples are discarded by the boosting method in both the CH and the DCH.

The training of BBSHR consists of two loops. The computational complexity for creating a hash function is $O(nd^2 + n_l^2 d)$ and the total computational complexity of the BBSHR is $O(mK(nd^2 + n_l^2 d))$ where n_l denotes the number of labeled images.

4.2. Parameter selection

There are two major parameters need to be selected for the proposed BBSHR, i.e. the bagging ratio (p) and the number of hash tables (m). Fig. 7 shows AUC performances of the BBSHR with different p values for the MNIST-32 using 5 hash tables. The value of p controls the sampling ratio of unlabeled samples in the construction of training sets for individual hash table learning. Fig. 7 shows that increasing the p value does not yield obvious effect to the performance of the BBSHR. In our experiments, $p = 0.4$ is used. This keeps a relatively large portion of unlabeled samples while provides a good trade-off with the computational costs.

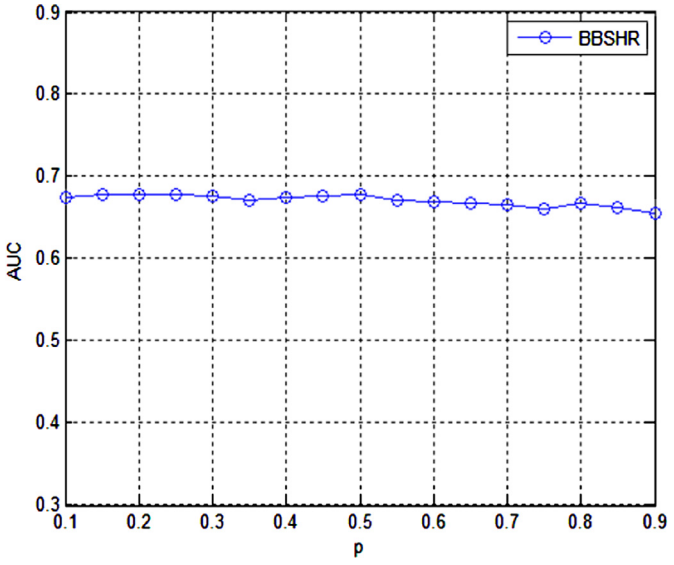

Fig. 7. AUC of the BBSHR method with different p values.

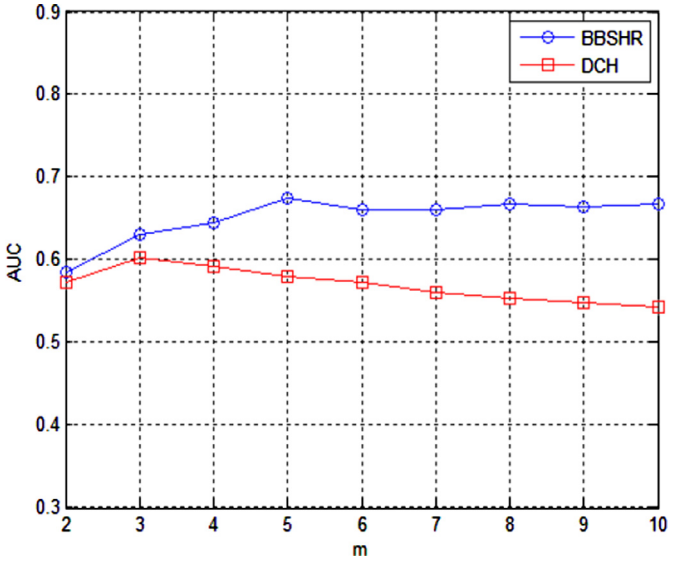

Fig. 8. AUC of the BBSHR method with different m values.

Fig. 8 shows AUC values of both the BBSHR and the DCH with different number of hash tables (m) using the MNIST-32 with $p = 0.4$. Fig. 8 shows that the performance of the BBSHR improves when m increases. However, the increment of m does not yield obvious influence to the performance of the BBSHR when $m > 5$. Therefore, $m = 5$ is used in our experiments. In contrast, the performance of the DCH decreases when $m > 3$. Again, the boosting of the DCH discards training samples when they are correctly classified by the current hash table, i.e. similar samples hashed to the same side of hash function. This makes the later hash tables (e.g. $m > 3$) have no useful training samples to learn. In the extreme case which all labeled samples are well learned and discarded, the hashing in later table reduces to unsupervised hashing.

5. Conclusion

A bagging–boosting-based semi-supervised multi-hashing method with query-adaptive re-ranking (BBSHR) is proposed in this paper. The BBSHR uses the semi-supervised bagging to construct multiple hash tables and then individual hash table is trained using a boosting-based method. The semi-supervised query-adaptive re-ranking is proposed to further improve retrieval performance. Experimental results show that the BBSHR outperforms state-of-the-art hashing methods with statistical significance. ann1 The current assignment method of pseudo-labels for unlabeled samples may not be optimal. In cases that nearest neighboring samples of an unlabeled sample are evenly distributed in several classes, the pseudo-label may require a random assignment among multiple majority classes. Further researches on a better pseudo-label assignment method may improve the performance of the BBSHR.

Acknowledgment

This work is under support of the National Natural Science Foundation of China under Grants (61272201 and 61572201) and the Fundamental Research Funds for the Central Universities (2017ZD052).

References

[1] C. Zhang, J.Y. Chai, R. Jin, User term feedback in interactive text-based image retrieval, in: Proceedings of the Twenty-eighth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 9, 2005, pp. 51–58. 1–3
[2] W. Li, L. Duan, D. Xu, I.W.-H. Tsang, Text-based image retrieval using progressive multi-instance learning, in: Proceedings of the 2011 International Conference on Computer Vision, 58, 2011, pp. 2049–2055.
[3] D. Petrelli, P. Clough, Using concept hierarchies in text-based image retrieval: a user evaluation, Lect. Notes Comput. Sci. 4022 (2005) 297–306.
[4] J. Zhao, Research on content-based multimedia information retrieval, in: Proceedings of the 2011 International Conference on Computational and Information Sciences, 2011, pp. 261–263.
[5] T. Kato, Cognitive View Mechanism for Content-Based Multimedia Information Retrieval, Springer, London, 1993, pp. 244–262.
[6] G. Zhou, M. Kai, F. Liu, Y. Yin, Relevance feature mapping for content-based multimedia information retrieval, Pattern Recognit. 45 (4) (2012) 1707–1720.
[7] C.S. Tong, M. Wong, Adaptive approximate nearest neighbor search for fractal image compression, IEEE Trans. Image Process. 11 (6) (2002) 605–615. A Publication of the IEEE Signal Processing Society.
[8] M. Casey, M. Slaney, Song intersection by approximate nearest neighbor search, in: Proceedings of the 2006 International Society for Music Information Retrieval (ISMIR), 6, 2006, pp. 144–149.
[9] P. Li, M. Wang, J. Cheng, C. Xu, H. Lu, Spectral hashing with semantically consistent graph for image indexing, IEEE Trans. Multimed. 15 (1) (2013) 141–152.
[10] J. Shao, F. Wu, C. Ouyang, X. Zhang, Sparse spectral hashing, Pattern Recognit. Lett. 33 (3) (2012) 271–277.
[11] W.W. Ng, Y. Lv, D.S. Yeung, P.P. Chan, Two-phase mapping hashing, Neurocomputing 151 (3) (2015) 1423–1429.
[12] B. Demir, L. Bruzzone, Hashing-based scalable remote sensing image search and retrieval in large archives, IEEE Trans. Geosci. Remote Sens. 54 (2) (2016) 892–904.
[13] X. Liu, Y. Mu, D. Zhang, B. Lang, X. Li, Large-scale unsupervised hashing with shared structure learning, IEEE Trans. Cybern. 45 (9) (2015) 1811–1822.
[14] L. Liu, L. Shao, Sequential compact code learning for unsupervised image hashing, IEEE Trans. Neural Netw. Learn. Syst. 27 (12) (2015) 2526–2536.
[15] L. Zhu, J. Shen, L. Xie, Unsupervised visual hashing with semantic assistant for content-based image retrieval, IEEE Trans. Knowl. Data Eng. 29 (2) (2016) 472–486.
[16] L. Liu, M. Yu, L. Shao, Unsupervised local feature hashing for image similarity search, IEEE Trans. Cybern. 46 (11) (2015) 2548–2558.
[17] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, S.-E. Yoon, Spherical hashing: binary code embedding with hyperspheres, IEEE Trans. Pattern Anal. Mach. Intell. 37 (11) (2015) 2304–2316.
[18] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: Proceedings of the 2009 International Conference on Very Large Data Bases, 2009, pp. 518–529.
[19] D. Gorisse, M. Cord, F. Precioso, Locality-sensitive hashing for chi2 distance, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2012) 402–409.
[20] Y. Matsushita, T. Wada, Principal component hashing: an accelerated approximate nearest neighbor search, Advances in Image and Video Technology, Springer, 2009, pp. 374–385.
[21] G. Yunchao, L. Svetlana, G. Albert, P. Florent, Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval, in: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, 35, 2013, pp. 2916–2929. 12
[22] C. Strecha, A. Bronstein, M. Bronstein, P. Fua, LDAHash: improved matching with smaller descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 34 (1) (2011) 66–78.

[23] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, S.-F. Chang, Supervised Hashing with Kernels, in: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2074–2081.
[24] V.-A. Nguyen, M.N. Do, Deep learning based supervised hashing for efficient image retrieval, in: Proceedings of the 2016 IEEE International Conference on Multimedia and Expo, 2016, pp. 1–6.
[25] J. Wang, S. Kumar, S.-F. Chang, Semi-supervised hashing for large-scale search, IEEE Trans. Pattern Anal. Mach. Intell. 34 (12) (2012) 2393–2406.
[26] C. Wu, J. Zhu, D. Cai, C. Chen, J. Bu, Semi-supervised nonlinear hashing using bootstrap sequential projection learning, IEEE Trans. Knowl. Data Eng. 25 (6) (2013) 1380–1393.
[27] C. Yao, J. Bu, C. Wu, G. Chen., Semi-supervised spectral hashing for fast similarity search, Neurocomputing 101 (2013) 52–58.
[28] W. Ng, Y. Lv, Z. Zeng, D. Yeung, P. Chan, Sequential conditional entropy maximization semi-supervised hashing for semantic image retrieval, in: Proceedings of the 2015 International Journal of Machine Learning and Cybernetics, 2015, pp. 1–16.
[29] L. Gao, J. Song, F. Zou, D. Zhang, J. Shao, Scalable multimedia retrieval by deep learning hashing with relative similarity learning, in: Proceedings of the Twenty-third ACM International Conference on Multimedia, 2015, pp. 903–906.
[30] H. Xie, Y. Zhang, J. Tan, L. Guo, J. Li, Contextual query expansion for image retrieval, IEEE Trans. Multimed. 16 (4) (2014) 1104–1114.
[31] L. Zhang, Y. Zhang, J. Tang, X. Gu, J. Li, Q. Tian, Topology preserving hashing for similarity search, in: Proceedings of the Twenty-first ACM International Conference on Multimedia, 2013, pp. 123–132.
[32] H. Xu, J. Wang, Z. Li, G. Zeng, Complementary hashing for approximate nearest neighbor search, in: Proceedings of the 2011 IEEE International Conference on Computer Vision, 2011, pp. 1631–1638.
[33] P. Li, J. Cheng, H. Lu, Hashing with dual complementary projection learning for fast image retrieval, Neurocomputing 120 (2013) 83–89.
[34] H. Fu, X. Kong, J. Lu, Large-scale image retrieval based on boosting iterative quantization hashing with query-adaptive reranking, Neurocomputing 122 (2013) 480–489.
[35] Z. Shaoting, Y. Ming, C. Timothee, Y. Kai, D. Metaxas, Query specific rank fusion for image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 37 (4) (2015) 803–815.
[36] J. Cheng, C. Leng, P. Li, M. Wang, H. Lu., Semi-supervised multi-graph hashing for scalable similarity search, Comput. Vis. Image Underst. 124 (2014) 12–21.
[37] L. Breiman, Using iterated bagging to Debias regressions, Mach. Learn. 45 (3) (2001) 261–277.
[38] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[39] J. Hulse, T. Khoshgoftaar, Experimental perspectives on learning from imbalanced data, in: Proceedings of the Twenty-Fourth International Conference on Machine Learning, 2008, pp. 935–942.
[40] H. Lai, P. Yan, X. Shu, Y. Wei, S. Yan, Instance-aware hashing for multi-label image retrieval, IEEE Trans. Image Process. 25 (6) (2016) 2469–2479.
[41] J. Myerson, L. Green, M. Warusawitharana, Area under the curve as a measure of discounting, J. Exp. Anal. Behav. 76 (2) (2001) 235–243.

Wing W. Y. Ng (S' 02-M' 05-SM' 15) received his B.Sc. and Ph.D. degrees from Hong Kong Polytechnic University in 2001 and 2006, respectively. He is now a Professor in the School of Computer Science and Engineering, South China University of Technology, China. His major research directions include machine learning and information retrieval. He is currently an associate editor of the International Journal of Machine Learning and Cybernetics. He is the principle investigator of three China National Nature Science Foundation projects and a Program for New Century Excellent Talents in University from China Ministry of Education. He served as the Board of Governor of IEEE Systems, Man and Cybernetics Society in 2011–2013.

Xiancheng Zhou received the B.Sc. and M.Sc. degrees in computer science from the South China University of Technology. His research interests include machine learning and information retrieval.

Xing Tian received his B.Sc. degree in Computer Science from the South China University of Technology, Guangzhou, China and is currently a Ph.D. candidate of the School of Computer Science and Engineering, South China University of Technology. His current research interests focus on image retrieval and machine learning in non-stationary big data environments.

Professor Xizhao Wang received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1998. He is currently a Professor with the Big Data Institute, Shenzhen University, Shenzhen, China. His current research interests include uncertainty modeling and machine learning for big data. He has edited more than ten special issues and published three monographs, two textbooks, and more than 200 peer-reviewed research papers. By the Google scholar, the total number of citations is over 5000. He is on the list of Elsevier 2015/2016 most cited Chinese authors. He is the Chair of the IEEE SMC Technical Committee on Computational Intelligence, the Editor-in-Chief of Machine Learning and Cybernetics Journal, and Associate Editor for a couple of journals in the related areas. He was a recipient of the IEEE SMCS Outstanding Contribution Award in 2004 and a recipient of the IEEE SMCS Best Associate Editor Award in 2006.

Professor Daniel S. Yeung (M' 89-SM' 99-F' 04) is a past President of the IEEE SMC Society. He was Head and Chair Professor of the Computing Department of Hong Kong Polytechnic University, Hong Kong, and a faculty member of Rochester Institute of Technology, USA. He has also worked for TRW Inc., General Electric Corporation R&D Centre and Computer Consoles Inc. in USA. He is a Fellow of the IEEE.