Rank Preserving Hashing for Rapid Image Search

Dongjin Song[*] Wei Liu[†] David A. Meyer[‡] Dacheng Tao[§] Rongrong Ji[♯]
[*]Department of ECE, UC San Diego, La Jolla, CA, USA
dosong@ucsd.edu
[†]IBM T. J. Watson Research Center, Yorktown Heights, New York, USA
weiliu@us.ibm.com
[‡]Department of Mathematics, UC San Diego, La Jolla, CA, USA
dmeyer@math.ucsd.edu
[§]Centre for Quantum Computation and Intelligent Systems
University of Technology, Sydney, Australia
dacheng.tao@uts.edu.au
[♯]Department of Cognitive Science, School of Information Science and Engineering
Xiamen University, Xiamen, P. R. China
rrji@xmu.edu.cn

Abstract

In recent years, hashing techniques are becoming overwhelmingly popular for their high efficiency in handling large-scale computer vision applications. It has been shown that hashing techniques which leverage supervised information can significantly enhance performance, and thus greatly benefit visual search tasks. Typically, a modern hashing method uses a set of hash functions to compress data samples into compact binary codes. However, few methods have developed hash functions to optimize the precision at the top of a ranking list based upon Hamming distances. In this paper, we propose a novel supervised hashing approach, namely Rank Preserving Hashing (RPH), to explicitly optimize the precision of Hamming distance ranking towards preserving the supervised rank information. The core idea is to train disciplined hash functions in which the mistakes at the top of a Hamming-distance ranking list are penalized more than those at the bottom. To find such hash functions, we relax the original discrete optimization objective to a continuous surrogate, and then design an online learning algorithm to efficiently optimize the surrogate objective. Empirical studies based upon two benchmark image datasets demonstrate that the proposed hashing approach achieves superior image search accuracy over the state-of-the-art approaches.

Introduction

With the rapidly growing scale and dimensionality of massive image collections such as Facebook, Instagram, and Flickr, how to search for visually relevant images effectively and efficiently has become an important problem. Instead of exhaustively searching for the most similar images with respect to a query in a high-dimensional feature space, hashing techniques encode images to compact binary codes via hash functions and perform efficient searches in the generated low-dimensional code space (*i.e.*, Hamming space), therefore drastically accelerating search procedures and also saving storage.

Conceptually, a modern hashing method exploits a set of hash functions $\big\{h_q : \mathbb{R}^d \mapsto \mathbb{H} = \{-1,1\}\big\}_{q=1}^{r}$ to map data samples from a d-dimensional data space $\mathbb{R}^d$ to an r-dimensional Hamming space $\mathbb{H}^r$, such that original samples are represented as compact binary hash codes of length r (typically less than 100). Early explorations in hashing, such as Locality-Sensitive Hashing (LSH) [1] and Min-wise Hashing (MinHash) [2], construct hash functions with random permutations or projections. These randomized hashing methods, however, require long code lengths ($r \geq 1,000$) to meet search requirements, and usually have inferior search accuracy for large-scale image search [3].

Unlike previous randomized hashing methods that are independent of data, a number of data-dependent hashing techniques have been invented in recent years. These techniques, in general, can be categorized into two main types: unsupervised and supervised (including semi-supervised) approaches. Unsupervised approaches, such as Spectral Hashing (SH) [4], Iterative Quantization (ITQ) [5], Isotropic Hashing (ISOH) [6], *etc.*, seek to learn hash functions by taking into account underlying data structures, distributions, or topological information. On the other hand, supervised approaches aim to learn hash functions by incorporating supervised information, *e.g.*, instance-level labels, pair-level labels, or triplet-level ranks. Representative techniques include Binary Reconstructive Embedding (BRE) [7], Minimal Loss Hashing (MLH) [8], Kernel-based Supervised Hashing (KSH) [3], Hamming Distance Metric Learning (HDML) [9], Ranking-based Supervised Hashing (RSH) [10], and Column Generation Hashing (CGH) [11].

Although the various supervised hashing techniques listed above have shown their effectiveness and efficiency for large-scale image search tasks, we argue that few of them explored optimizing the precision at the top of a ranking list according to Hamming distances among the generated hash codes, which is indeed one of the most practical concerns with high-performance image search. To this end, in this paper we propose a rank preserving hashing technique specifically designed for optimizing the precision of Hamming distance ranking. The core idea is to train disciplined hash functions in which the mistakes at the top of a Hamming-distance ranking list are penalized more than those at the bottom. Since the rank-preserving objective we introduced is discrete in nature and the associated optimization problem is combinatorially difficult, we relax the original discrete objective to a continuous and differentiable surrogate, and then design an online learning algorithm to efficiently optimize the surrogate objective. We compare the proposed approach, dubbed *Rank Preserving Hashing* (RPH), against various state-of-the-art hashing methods through extensive experiments conducted on two benchmark image datasets, CIFAR10 [12] and SUN397 [13]. The experimental results demonstrate that RPH outperforms the state-of-the-art approaches in executing rapid image search with binary hash codes.

The rest of this paper is organized as follows. In Section 2, we describe the learning model of RPH. In Section 3, we study a surrogate optimization objective of RPH, and accordingly design an online learning algorithm to tackle this objective. In Section 4, we report and analyze the experimental results. Finally, we conclude this work in Section 5.

Rank Preserving Hashing

In this section, we first introduce some main notations used in the paper. Let $\mathbf{X} \in \mathbb{R}^{d\times n}$ be a data matrix of n data samples with d dimensions, $\boldsymbol{x}_i \in \mathbb{R}^d$ be the i-th column of $\mathbf{X}$, and X_{ij} be the entry in i-th row and j-th column of $\mathbf{X}$, respectively. Moreover, we use $\|\cdot\|_{\mathrm{F}}$ to denote the Frobenius norm of matrices, and $\|\boldsymbol{x}\|_{\mathrm{H}}$ to represent the Hamming norm of vector $\boldsymbol{x}$, which is defined as the number of nonzero entries in $\boldsymbol{x}$, *i.e.*, ℓ_0 norm. We use $\|\boldsymbol{x}\|_1$ to represent the ℓ_1 norm of vector $\boldsymbol{x}$, which is defined as the sum of absolute values of the entries in $\boldsymbol{x}$.

Given a data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, the purpose of our proposed supervised hashing model RPH is to learn a set of mapping functions $\big\{h_q(\boldsymbol{x})\big\}_{q=1}^{r}$ such that a d-dimensional floating-point input $\boldsymbol{x} \in \mathbb{R}^d$ is compressed into an r-bit binary code $\boldsymbol{b} = \big[h_1(\boldsymbol{x}), \cdots, h_r(\boldsymbol{x})\big]^\top \in \mathbb{H}^r \equiv \{1, -1\}^r$. This mapping, known as hash function in the literature, is formulated by

$$h_q(\boldsymbol{x}) = \mathrm{sgn}\big(f_q(\boldsymbol{x})\big), \quad q = 1, \cdots, r, \tag{1}$$

where $\mathrm{sgn}(x)$ is the sign function that returns 1 if input variable $x > 0$ and -1 otherwise, and $f_q : \mathbb{R}^d \mapsto \mathbb{R}$ is a proper prediction function. A variety of mathematical forms for f_q (*e.g.*, linear or nonlinear) can be utilized to serve to specific data domains and practical applications. In this work, we focus on using a linear prediction function, that is, $f_q(\boldsymbol{x}) = \boldsymbol{w}_q^\top \boldsymbol{x} + t_q$ (where $\boldsymbol{w}_q \in \mathbb{R}^d$ and $t_q \in \mathbb{R}$) for simplicity. Following the previous work [5][6][3][10], we set the bias term $t_q = -\boldsymbol{w}_q^\top \boldsymbol{u}$ by using the mean vector $\boldsymbol{u} = \sum_{i=1}^n \boldsymbol{x}_i/n$, which will make each generated hash bit $\big\{h_q(\boldsymbol{x}_i)\big\}_{i=1}^n$ for $q \in [1:r]$ be nearly balanced and hence exhibit maximum entropy. For brevity, we further define a coding function $\boldsymbol{h} : \mathbb{R}^d \mapsto \mathbb{H}^r$ to comprise the functionality of r hash functions $\{h_q\}_q$, that is,

$$\boldsymbol{h}(\boldsymbol{x}, \mathbf{W}) = \mathrm{sgn}\big(\mathbf{W}^\top(\boldsymbol{x} - \boldsymbol{u})\big), \tag{2}$$

which is parameterized by a matrix $\mathbf{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_r] \in \mathbb{R}^{d\times r}$. Note that Eq. (2) applies the sign function $\mathrm{sgn}(\cdot)$ in the element-wise way.

To pursue such a coding function $\boldsymbol{h}(\cdot)$, rather than considering pairwise data similarities (*i.e.*, pair-level labels) as in [3], we leverage relative data similarities in the form of triplets $\mathcal{D} = \big\{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s)\big\}$, in which the pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is more similar than the pair $(\boldsymbol{x}_i, \boldsymbol{x}_s)$. Intuitively, we would expect that these relative similar relationships revealed by $\mathcal{D}$ can be preserved within the Hamming space by virtue of a good coding function $\boldsymbol{h}(\cdot)$, which makes the Hamming distance between the codes $\boldsymbol{h}(\boldsymbol{x}_i, \mathbf{W})$ and $\boldsymbol{h}(\boldsymbol{x}_j, \mathbf{W})$ smaller than that between the codes $\boldsymbol{h}(\boldsymbol{x}_i, \mathbf{W})$ and $\boldsymbol{h}(\boldsymbol{x}_s, \mathbf{W})$. Suppose that $\boldsymbol{x}_i$ is a query, $\boldsymbol{x}_j$ is its similar sample, and $\boldsymbol{x}_s$ is its dissimilar sample. Then we define the "rank" of $\boldsymbol{x}_j$ with respect to the query $\boldsymbol{x}_i$ as:

$$\mathrm{R}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_s I\left(\big\|\boldsymbol{h}(\boldsymbol{x}_i, \mathbf{W}) - \boldsymbol{h}(\boldsymbol{x}_s, \mathbf{W})\big\|_{\mathrm{H}} \le \big\|\boldsymbol{h}(\boldsymbol{x}_i, \mathbf{W}) - \boldsymbol{h}(\boldsymbol{x}_j, \mathbf{W})\big\|_{\mathrm{H}}\right), \tag{3}$$

where $I(\cdot)$ is an indicator function which returns 1 if the condition in the parenthesis is satisfied and returns 0 otherwise. The function $\mathrm{R}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ measures the number of the incorrectly ranked negative samples $\boldsymbol{x}_s$'s which are closer to the query $\boldsymbol{x}_i$ than the positive sample $\boldsymbol{x}_j$ in terms of Hamming distance. In order to explicitly optimize the search precision at top positions of a ranking list, we introduce a ranking loss

$$L\big(\mathrm{R}(\boldsymbol{x}_i,\boldsymbol{x}_j)\big)=\sum_{c=1}^{\mathrm{R}(\boldsymbol{x}_i,\boldsymbol{x}_j)}\frac{1}{c} \tag{4}$$

which penalizes the samples that are incorrectly ranked to be at top positions. Our model RPH makes use of the above ranking loss to optimize the search precision, whose learning objective is formulated as follows:

$$\mathcal{O}(\mathbf{X},\mathbf{W})=\sum_i\sum_j L\big(\mathrm{R}(\boldsymbol{x}_i,\boldsymbol{x}_j)\big)+\frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{5}$$

where the second term enforces regularization, and $\lambda > 0$ is a positive parameter controlling the trade-off between the ranking loss and the regularization term.

We are aware that the ranking loss in Eq. (4) were previously used for information retrieval [14] and image annotation [15]. Our work differs from the previous ones because the relative similarity of the triplets are measured with Hamming distance which is discrete and discontinuous.

Optimization

The objective of RPH in Eq. (5) is difficult to optimize. This is because: (a) the hash function is a discrete mapping; (b) the Hamming norm lies in a discrete space; and (c) the ranking loss is piecewise constant. These reasons result in a discontinuous and non-convex objective (Eq. (5)) which is combinatorially difficult to optimize.

To tackle this issue, we need to relax the objective so that it can be continuous differentiable.

Relaxation

Specifically, the hash function $\boldsymbol{h}(\boldsymbol{x},\mathbf{W})=\mathrm{sgn}\big(\mathbf{W}^\top(\boldsymbol{x}-\boldsymbol{u})\big)$ can be relaxed as:

$$\overline{\boldsymbol{h}}(\boldsymbol{x},\mathbf{W})=\tanh\big(\mathbf{W}^\top(\boldsymbol{x}-\boldsymbol{u})\big), \tag{6}$$

which is continuous and differentiable as shown in Figure 1(a). $\tanh(\cdot)$ is a good approximation for $\mathrm{sgn}(\cdot)$ function because it transforms the value in the parenthesis to be in between of -1 and $+1$.

Next, the Hamming norm in Eq. (3) is relaxed to ℓ_1 norm which is convex.

Finally, we relax the indicator function in Eq. (3) with hinge loss which is a convex upper bound of Eq. (3) and can be represented as:

$$\begin{aligned}&I\Big(\big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i,\mathbf{W})-\overline{\boldsymbol{h}}(\boldsymbol{x}_s,\mathbf{W})\big\|_1\le\big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i,\mathbf{W})-\overline{\boldsymbol{h}}(\boldsymbol{x}_j,\mathbf{W})\big\|_1\Big)\\ \le&\Big|1-\big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i,\mathbf{W})-\overline{\boldsymbol{h}}(\boldsymbol{x}_s,\mathbf{W})\big\|_1+\big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i,\mathbf{W})-\overline{\boldsymbol{h}}(\boldsymbol{x}_j,\mathbf{W})\big\|_1\Big|_+,\end{aligned} \tag{7}$$

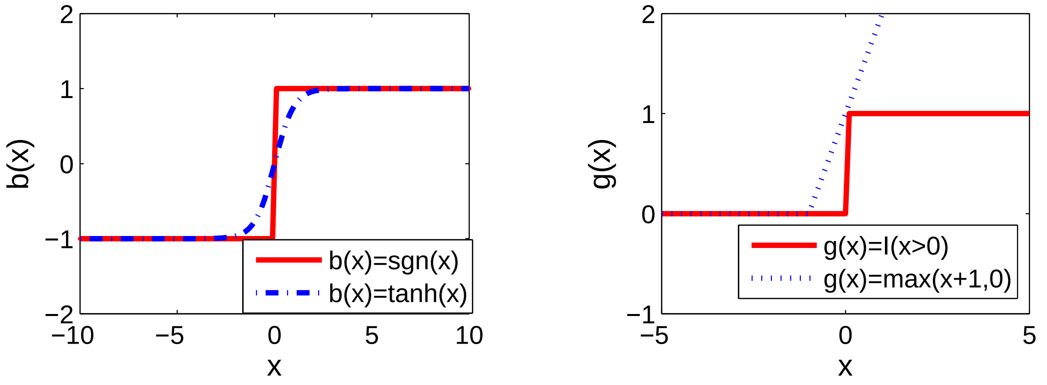


Figure 1: Relaxation of the objective function. (a) $\tanh(x)$ is a relaxation of $\mathrm{sgn}(x)$; (b) hinge loss $g(x) = \max(x+1, 0)$ is a convex upper bound of indicator function $g(x) = I(x > 0)$.

where $|t|_+ = \max(0, t)$.

Based upon these relaxations, the objective in Eq. (5) can be approximated as:

$$\begin{aligned}\overline{\mathcal{O}(\mathbf{X}, \mathbf{W})} =& \lambda\|\mathbf{W}\|_{\mathrm{F}}^2 + \sum_i \sum_j L(\overline{\mathrm{R}}(\boldsymbol{x}_i, \boldsymbol{x}_j))\cdot \\ & \frac{\left|1 - \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W})\right\|_1 + \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W})\right\|_1\right|_+}{\overline{\mathrm{R}}(\boldsymbol{x}_i, \boldsymbol{x}_j)}\end{aligned} \tag{8}$$

where $\overline{\mathrm{R}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the margin-penalized rank of $\boldsymbol{x}_j$ which is given by:

$$\overline{\mathrm{R}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_s I\Big(1 + \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W})\right\|_1 \geq \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W})\right\|_1\Big). \tag{9}$$

Although full gradient descent approach can be derived to optimize Eq. (8), it may converge slowly or even will be infeasible because of the expensive computational for the gradient at each iteration. Therefore, an online learning method is derived to resolve this issue.

Online learning

For online learning, we consider a triplets $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s)$ in which the pair of $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are more similar than the pair of $(\boldsymbol{x}_i, \boldsymbol{x}_s)$, and then uniformly sample $\boldsymbol{x}_s$, $s = 1, ..., p$ ($p \leq |\mathcal{N}|$, where $|\mathcal{N}|$ is the total number of possible choices of s) when i and j are fixed until we find a violation which satisfies $1 + \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W})\right\|_1 > \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W})\right\|_1$. In this way, the objective of the chosen triplets can be given by:

$$\begin{aligned}\overline{\mathcal{O}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s, \mathbf{W})} =& \frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 + L(\overline{\mathrm{R}}(\boldsymbol{x}_i, \boldsymbol{x}_j))\cdot \\ & \left|1 - \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W})\right\|_1 + \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W})\right\|_1\right|_+.\end{aligned} \tag{10}$$

Algorithm 1: Online learning for Rank Preserving Hashing (RPH)

```
1: Input: data $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$, label $\boldsymbol{y} = [y_1; y_2; ...; y_n] \in \mathbb{R}^n$, $\alpha$, and $\lambda$
2: Output: $\mathbf{W} \in \mathbb{R}^{d\times r}$
3: repeat
4:    Pick a random triplets ($\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_s$).
5:    Calculate $\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W})$, $\overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W})$, and $\overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W})$
6:    Set p=0
7:       repeat
8:          Fix $i$ and $j$ and pick a random triplets ($\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_s$) when $s$ varies.
9:          Calculate $\overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W})$
10:         $p = p + 1$
11:      until $1 + \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W})\right\|_1 > \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W})\right\|_1$ or $p \geq |\mathcal{N}|$
12:      if $1 + \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W})\right\|_1 > \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W})\right\|_1$, then
13:         Calculate the gradient in Eq. (12)
14:         Make a gradient descent based upon Eq. (11)
15:      end if
16: until validation error does not improve or maximum iteration number is achieved.
```

Assuming that the violated examples are uniformly distributed, then $\overline{\mathrm{R}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be approximated with $\lfloor \frac{|\mathcal{N}|}{p} \rfloor$ where $\lfloor \cdot \rfloor$ is the floor function. Therefore, we can perform the following stochastic update procedure over the parameter $\mathbf{W}$, which is given as:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha \frac{\partial \overline{\mathcal{O}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s, \mathbf{W})}}{\partial \mathbf{W}_t}, \tag{11}$$

where α is the learning rate and the gradient is given by:

$$\begin{aligned}\frac{\partial \overline{\mathcal{O}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s, \mathbf{W})}}{\partial \mathbf{W}_t} =& \lambda \mathbf{W}_t + L\Big(\Big\lfloor \frac{|\mathcal{N}|}{p} \Big\rfloor\Big) \cdot \Big\{ \\ & (\boldsymbol{x}_i - \boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}_t) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W}_t)\Big) \bigodot \Big(1 - \overline{\boldsymbol{h}}^2(\boldsymbol{x}_i, \mathbf{W}_t)\Big)\Big]^\top - \\ & (\boldsymbol{x}_j - \boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}_t) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j, \mathbf{W}_t)\Big) \bigodot \Big(1 - \overline{\boldsymbol{h}}^2(\boldsymbol{x}_j, \mathbf{W}_t)\Big)\Big]^\top - \\ & (\boldsymbol{x}_i - \boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}_t) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W}_t) \bigodot \Big(1 - \overline{\boldsymbol{h}}^2(\boldsymbol{x}_i, \mathbf{W}_t)\Big)\Big]^\top + \\ & (\boldsymbol{x}_s - \boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W}_t) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s, \mathbf{W}_t) \bigodot \Big(1 - \overline{\boldsymbol{h}}^2(\boldsymbol{x}_s, \mathbf{W}_t)\Big)\Big]^\top)\Big\},\end{aligned} \tag{12}$$

where $\bigodot$ denote element-wise product.

The detailed online learning procedure for RPH is shown in Algorithm 1. The main computation is the gradient updating and the associated computational complexity is $O(d^2r + drp)$. Therefore, the computation is feasible at each step. After getting $\mathbf{W}$, the compact binary codes for each sample can be obtained with Eq. (2).

Experiments

In this section, we first introduce two publicly available datasets for empirical study. Then we provide the experimental setup and the evaluation metrics used in our experiment. Finally, we compare our proposed Rank Preserving Hashing (RPH) against baseline algorithms.

Datasets

In our experiment, we conduct image search over two datasets, *i.e.*, CIFAR10 [12] and SUN397 [13]. CIFAR10 is a labeled subset of 80 Million Tiny Image dataset [16]. It contains 60K images from ten object categories with each image represented by a 512-dimensional GIST feature vector. SUN397 consists of around 108K images of 397 scenes. In SUN397, each image is represented by a 1600-dimensional feature vector extracted by principle component analysis (PCA) from 12288 dimensional Deep Convolutional Activation Features [17].

Experimental Setup

Based upon these two datasets, we compare the proposed RPH against seven representative hashing techniques. Four of them are unsupervised approaches, including one randomized method, Locality Sensitive Hashing (LSH) [1], one spectral approach, Spectral Hashing (SH) [4], and two linear projection techniques, Iterative Quantization (ITQ) and Isotropic Hashing (ISOH). The other three are supervised approaches which use triplets to encode the label information (similar to our setting); they are Hamming Distance Metric Learning (HDML) [9], Column Generation Hashing (CGH) [11], and Ranking-based Supervised Hashing (RSH) [10].

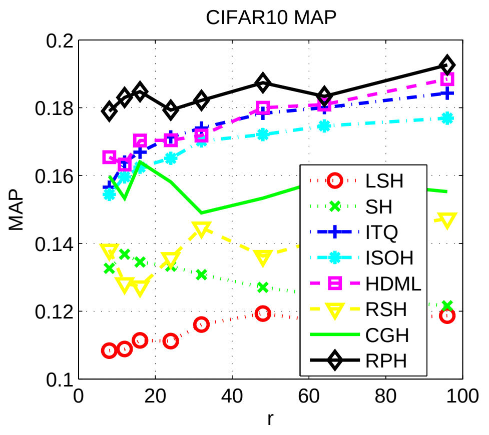
In CIFAR10, we randomly sample 100 images from each object category to form a separate test set of 1000 images as the queries. For supervised learning, we randomly select 400 additional images from each object category to form the training set and 50 images from each object category to be the validation set (query). The rest of the images are used as the database for testing.

Similarly, in SUN397, 100 images are randomly sampled from each of the 18 largest scene categories to form a test (query) set of 1800 images. We also randomly select 200 additional images from each of these large scene categories to be the training set of 3600 images and 50 additional images from each of them to form the validation set (query). The rest of the images are also used as the database for testing.

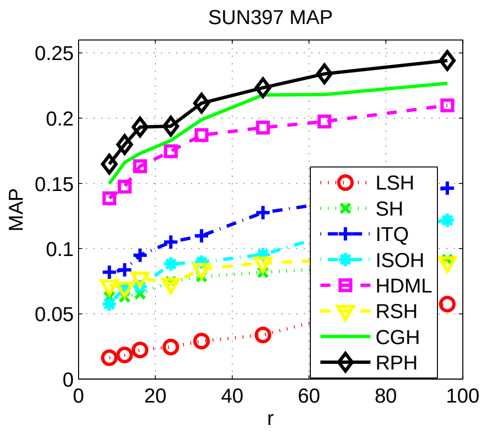
Evaluation Metrics

To measure the effectiveness of various hashing techniques for ranking, we consider three evaluation metrics, *i.e.*, Mean Average Precision (MAP), precision at top-k positions (P@k), and recall at top-k positions (Recall@k). In particular, P@k is defined as:

$$P@k = \frac{\#\textit{positive samples in the top } k}{\#\textit{positive and negative samples in the top } k},$$



(a) MAP over CIFAR10



(b) MAP over SUN397

Figure 2: MAP versus number of hash bits (*i.e.*, $r = \{8, 12, 16, 24, 32, 48, 64, 96\}$) for different hashing techniques over CIFAR10 and SUN397.

and Recall@k is given by:

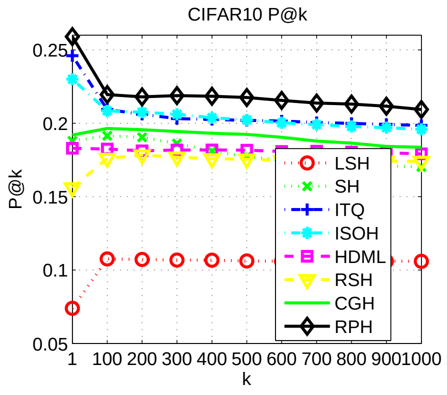
$$Recall@k = \frac{\#positive\ samples\ in\ the\ top\ k}{\#positive\ positive\ examples}.$$

Results

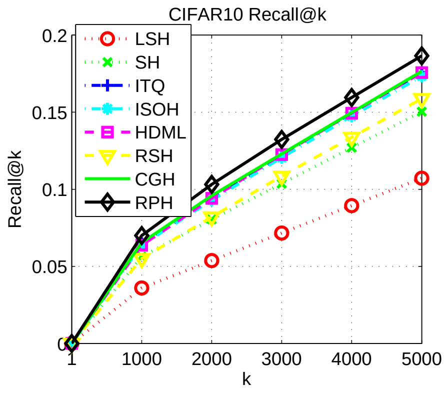
Figure 2 compares the MAP of the proposed RPH with baseline approaches based upon CIFAR10 and SUN397 when the number of hash bits varies. Clearly, RPH outperforms baseline approaches in most of the cases. This is because the objective of RPH is specifically designed to optimize the top-k precision by penalizing the mistakes at the top of a ranking list more than the mistakes at the bottom and thus can preserve the supervised rank information. For the baseline approaches, supervised approaches (e.g., HDML, RSH, and CGH), in general, outperform LSH since they leverage label information to learn discriminative hash functions for binary code generation. Among the unsupervised approaches, SH, ITQ, and ISOH consistently outperform LSH because they considered the underlying data structures, distributions, or topological information rather than producing the hash functions randomly.

We further investigate the effectiveness of the proposed RPH by comparing its P@k and Recall@k with baseline methods over two databases in Figure 3 and 4, respectively. We observe that, with the position k increasing, the P@k and Recall@k of RPH consistently outperform other approaches over these two datasets when the number of bits is fixed as 8 and 96, respectively. This indicates that modeling the supervised rank information appropriately can significantly improve the top-k image search accuracy.

In contrast to the offline training time, the online code generation time is more crucial for image search applications. In term of code generation, the main computation for RPH depends on the linear projection and binarization. Therefore, RPH's code generation is as efficient as LSH, ITQ, and ISOH.

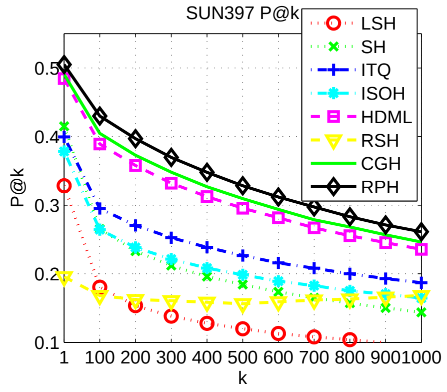


(a) P@k with 8 bits on CIFAR10

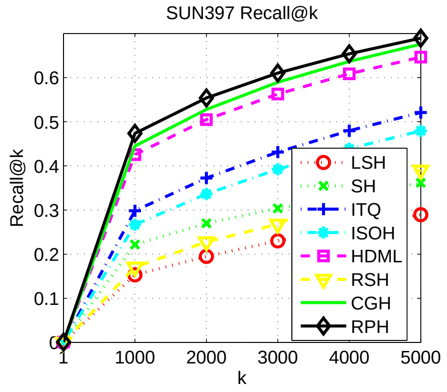


(b) Recall@k with 8 bits on CIFAR10

Figure 3: P@k and Recall@k for CIFAR10 with 8 bits hash codes.



(a) P@k with 96 bits on SUN397



(b) Recall@k with 96 bits on SUN397

Figure 4: P@k and Recall@k for SUN397 with 96 bits hash codes.

Conclusion

In this paper, we proposed a novel supervised hashing technique, dubbed Rank Preserving Hashing (RPH), to support rapid image search. Unlike previous supervised hashing methods, the proposed RPH aims to learn disciplined hash functions through explicitly optimizing the precision at the top positions of Hamming-distance ranking lists, thereby preserving the supervised rank information. Since the objective of RPH is discrete and the associated optimization problem is combinatorially difficult, we relaxed the original discrete objective to a continuous surrogate, and then devised an online learning algorithm to efficiently optimize the surrogate objective. We conducted extensive experiments on two benchmark image datasets, demonstrating that RPH outperforms the state-of-the-art hashing methods in terms of executing image search with compact binary hash codes.

Acknowledgement

Dongjin Song and David A. Meyer were partially supported by the Minerva Research Initiative under ARO grants W911NF-09-1-0081 and W911NF-12-1-0389. Dacheng Tao was supported by Australian Research Council Projects DP-140102164, FT-

130101457, and LP-140100569. Rongrong Ji was supported by the National Natural Science Foundation of China Under Grant Number 61422210 and 61373076.

References

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

[2] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permuations," in *Proceedings of ACM Symposium on Theory of Computing*, 1998.

[3] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2012.

[4] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proceedings of Advances in Neural Information Processing Systems 21*, 2008.

[5] 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 Interlligence*, 2012.

[6] W. Kong and W.-J. Li, "Istropic hashing," in *Proceedings of Advances in Neural Information Processing Systems 25*, 2012.

[7] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proceedings of Advances in Neural Information Processing Systems 22*, 2009.

[8] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[9] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *Proceedings of Advances in Neural Information Processing Systems 25*, 2012.

[10] J. Wang, W. Liu, A. X. Sun, and Y. Jiang, "Learning hash codes with listwise supervision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[11] X. Li, G. Lin, C. Shen, A. Hengel, and A. Dick, "Learning hash functions using column generation," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[12] A. Krizhevsky, "Learning multiple layers of features from tiny images," in *Technical report*, 2009.

[13] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2010.

[14] N. Usunier, D. Buffoni, and P. Gallinari, "Ranking with ordered weighted pairwise classification," in *Proceedings of the 26th International Conference on Machine Learning*, 2009, pp. 1057–1064.

[15] J. Weston, S. Bengio, and N. Usunier, "Large scale image annotation: learning to rank with joint world-image embeddings," *Machine Learning*, vol. 81, no. 1, pp. 21–35, 2012.

[16] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large databases for recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[17] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proceedings of European Conference on Computer Vision*, 2014.