Large-Margin Supervised Hashing

Xiaopeng Zhang, Hui Zhang, Yong Chen(✉), and Xianglong Liu

State Key Lab of Software Development Environment,
Beihang University, Beijing, China
{zxpjustin,hzhang}@buaa.edu.cn, {chenyong,xlliu}@nlsde.buaa.edu.cn

Abstract. Learning to hash embeds objects (e.g. images/documents) into a binary space with the semantic similarities preserved from the original space, which definitely benefits large-scale tough tasks such as image retrieval. By leveraging semantic labels, supervised hashing methods usually achieve better performance than unsupervised ones in real-world scenarios. However, most existing supervised methods do not sufficiently encourage inter-class separability and intra-class compactness which is quite crucial in discriminative hashcodes. In this paper, we propose a novel hashing method called Large-Margin Supervised Hashing (LMSH) based on a non-linear classification framework. Specifically, LMSH introduces the angular decision margin which could adjust inter-class separability and intra-class compactness through a hyper-parameter for more discriminative codes. Extensive experiments on three public datasets are conducted to demonstrate the LMSH's superior performance to some state-of-the-arts in image retrieval tasks.

Keywords: Large-margin · Supervised hashing · Non-linear classification framework

1 Introduction

Recently, hashing techniques, as a most popular candidate for approximate nearest neighbor search, have been widely used for lots of practical problems, such as speech recognition, information retrieval, computer vision, and nature language processing. More specifically, learning to hash can transform images, documents and videos to compact binary representations while simultaneously preserving the similarities of the original data with hamming distances. Hashing representations have two manifest advantages: (1) binary hash codes need less storage space; (2) search can be performed in sublinear time by computing Hamming distance (XOR operation) or in near O(1) time with hash tables.

Generally speaking, current hashing methods can be divided into two groups: the unsupervised methods and the supervised methods. The methods which do not rely on labeled data are classified into unsupervised hashing methods, such as LSH [6], SpH [23], DGH [12], ITQ [7], BSH [20], BS [18], CH [17], SGH [8], MVCH [19] and SSH [21]. The other category is supervised hashing methods which usually attain higher retrieval accuracy, since the label information has

D. Liu et al. (Eds.): ICONIP 2017, Part I, LNCS 10634, pp. 259–269, 2017.
https://doi.org/10.1007/978-3-319-70087-8_28

been taken full advantage of, such as BRE [10], KSH [13], FashHash [5], LFH [24], SDH [22], TSH [11], COSDISH [9], MH [16], and RMTHL [2].

We are mainly concerned about supervised hashing methods. Because of the difficulty of discrete optimization problem, most supervised hashing methods solve a relaxed continuous optimization problem by dropping the discrete constraints. However, these methods can't ignore the quantization error of mapping continuous data to binary space. To solve this problem, some methods try to directly solve the discrete optimization problem [9,12,22], and they both make progress and meet some specific problems. Meanwhile, this paper present an another new idea, which is easy to accomplish, to reduce the quantization error.

We propose a novel supervised approach called Large-Margin Supervised Hashing (LMSH) based on a non-linear classification framework. Notably, our LMSH combines a large angular decision margin which both improves the classification generalization and leads to less quantization error. In brief, our contributions can be summarized as below:

- A novel supervised hashing method is presented with the perspective of large angular decision margin, which owns a clear geometric interpretation and encourages the inter-class separability and intra-class compactness for more discriminative codes.
- The large angular decision margin proposed is a new tool to reduce the quantization error in hashing task. And the size of the angular decision margin can be flexibly adjusted with a preset constant m, which is a trade-off parameter between efficiency and efficacy.
- The proposed LMSH is evaluated on three large image benchmarks and exhibits superior retrieval performance to many state-of-the-art hashing methods.

2 Preliminary

Supposed that we have N samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ where each sample $\mathbf{x}_i$ is a M-dimensional vector. The corresponding ground truth labels are represented as $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$, and each $\mathbf{y}_i$ is a C-dimensional vector for sample $\mathbf{x}_i$, where C expresses the number of label classes and $y_{ci} = 1$ if $\mathbf{x}_i$ belongs to class c and 0 otherwise. The goal of hashing methods is to learn hash functions transforming $\mathbf{X}$ into a binary code $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^N \in \{-1, 1\}^{K \times N}$ which would preserve the sematic similarities. The vector $\mathbf{b}_i$, which denotes the i^{th} column of $\mathbf{B}$, is the K-bits binary codes for sample $\mathbf{x}_i$.

Inspired by SDH [22], we firstly consider the hashing code learning in the framework of linear classification to take advantage of the supervised information. We assume that good binary codes also benefit the classification task. For the i^{th} sample, we employ the following multi-class classification formulation:

$$\hat{\mathbf{y}}_i = \mathbf{W}^{\mathrm{T}}\mathbf{b}_i = [\mathbf{w}_1^{\mathrm{T}}\mathbf{b}_i, \cdots, \mathbf{w}_C^{\mathrm{T}}\mathbf{b}_i]^{\mathrm{T}}, \tag{1}$$

where $\mathbf{w}_c \in \mathbb{R}^{K \times 1} (c = 1, \cdots, C)$ is the classification vector for class c, and $\hat{\mathbf{y}} \in \mathbb{R}^{C \times 1}$ is the predicted classes vector regarding to $\mathbf{b}_i$, of which the maximum

item denotes the assigned class of $\mathbf{x}_i$. The difference between $\hat{\mathbf{y}}_i$ and the true label vector $\mathbf{y}_i$ is expected to be as small as possible. Therefore, we should optimize the following problem with ℓ_2 loss function:

$$\min_{\mathbf{B},\mathbf{W}} \quad \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{W}^{\mathrm{T}}\mathbf{b}_i\|^2 + \lambda\|\mathbf{W}\|^2, \quad s.t. \quad \mathbf{b}_i \in \{-1, 1\}^K, \tag{2}$$

where $\|\cdot\|$ represents Frobenius norm for matrices and ℓ_2 norm for vectors, and λ is a non-negative hyper-parameter to avoid overfitting.

Then we can solve problem (2) by updating $\mathbf{W}$ and $\mathbf{B}$ alternately until convergence as follows.

W-Step. With $\mathbf{B}$ fixed, we rewrite the optimization problem (2) as:

$$\min_{\mathbf{W}} \quad \|\mathbf{Y} - \mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 + \lambda\|\mathbf{W}\|^2. \tag{3}$$

Then we can easily update $\mathbf{W}$ by the following equation:

$$\mathbf{W} = (\mathbf{B}\mathbf{B}^{\mathrm{T}} + 2\lambda\mathbf{I})^{-1}\mathbf{B}\mathbf{Y}^{\mathrm{T}}, \tag{4}$$

where $\mathbf{I}$ is an identity matrix.

B-Step. Optimizing $\mathbf{B}$ with $\mathbf{W}$ fixed is still NP hard and difficult to solve directly owing to the discrete constraints. Following most existing schemes, we update $\mathbf{B}$ through two stages. Firstly, we relax $\mathbf{B}$ to a real matrix $\mathbf{V} \in \mathbb{R}^{K\times N}$:

$$\min_{\mathbf{V}} \quad \|\mathbf{Y} - \mathbf{W}^{\mathrm{T}}\mathbf{V}\|^2, \tag{5}$$

and then optimize $\mathbf{V}$ in the real space:

$$\mathbf{V} = (\mathbf{W}\mathbf{W}^{\mathrm{T}})^{-1}\mathbf{W}\mathbf{Y}. \tag{6}$$

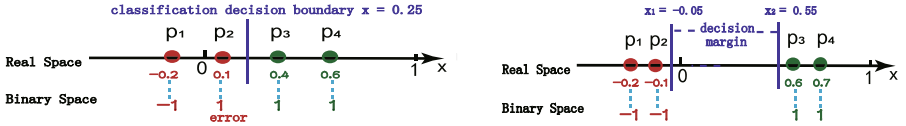
Secondly, a simple rounding technique can be performed to project the real valued $\mathbf{V}$ to the binary matrix $\mathbf{B}$: $\mathbf{B} = sgn(\mathbf{V})$, where $sgn(\cdot)$ is the sign function, which outputs 1 for positive numbers and -1 otherwise (0 is rounding threshold).

In summary, we can carry out W-Step and B-Step alternately until the convergence is reached. However, the performance obtained is unsatisfactory, which would be ascribed to much quantization error.

As shown in Fig. 1(a), even though each points are classified correctly by a perfect classifier, we will possibly obtain fault codes. The intrinsical reason for this error is that we can't guarantee the consistency of the classification boundary and the rounding threshold. However, if the decision boundary is turned into a wide decision margin as Fig. 1(b), the quantization error will be reduced.

3 The Proposed Method: LMSH

Angular Decision Margin: To obtain a large decision margin, we introduce a stronger classification criterion. Inspired by [15], we replace $\mathbf{w}_c^{\mathrm{T}}\mathbf{b}$ with



(a) Original algorithm (b) Large-margin algorithm

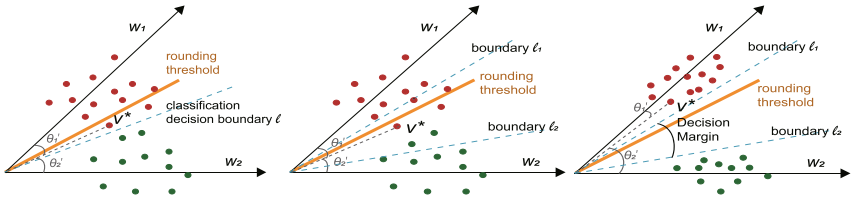
Fig. 1. Examples to show the effect of the large decision margin in 1-D space. Points with same color are in the same class. The default rounding threshold is $x = 0$.

$\|\mathbf{w}_c\|\|\mathbf{b}\|\cos(m\theta_c)$, where m is a positive integer, and $\mathbf{w}_c$ and $\mathbf{b}$ are the column vectors of $\mathbf{W}$ and $\mathbf{B}$ in problem (3) respectively, and c is the class index. The θ_c is the angle between $\mathbf{w}_c$ and $\mathbf{b}$.

To describe our intuition, a simple example is provided where all points belong to class 1 or class 2. Suppose $\mathbf{b}$ is the hash code of a sample $\mathbf{x}$ labeled by class 1. The original algorithm just needs to force $\mathbf{w}_1^{\mathrm{T}}\mathbf{b} > \mathbf{w}_2^{\mathrm{T}}\mathbf{b}$ in order to classify $\mathbf{b}$ correctly. However, to make a larger decision margin, we require:

$$\|\mathbf{w}_1\|\|\mathbf{b}\|\cos(m\theta_1) > \|\mathbf{w}_2\|\|\mathbf{b}\|\cos(\theta_2) \quad \left(0 \le \theta_1 \le \frac{\pi}{m}\right), \tag{7}$$

where m is a positive integer. Since $\cos(\cdot)$ is a monotone decreasing function on $[0, \pi]$ and $m \ge 1$, $\|\mathbf{w}_1\|\|\mathbf{b}\|\cos(\theta_1) \ge \|\mathbf{w}_1\|\|\mathbf{b}\|\cos(m\theta_1)$ always holds on $[0, \frac{\pi}{m}]$. If the Eq. (7) held, $\mathbf{w}_1^{\mathrm{T}}\mathbf{b} > \mathbf{w}_2^{\mathrm{T}}\mathbf{b}$ would be bound to hold, too. Therefore the Eq. (7) is a stronger requirement to classify $\mathbf{b}$ correctly, resulting in a larger decision margin between class 1 and class 2.



(a) The original method (converged) (b) Large-margin method (not converged) (c) Large-margin method (converged)

Fig. 2. The geometric difference between the original algorithm and the large-margin extensions. The red points belong to class 1, and the greens belong to class 2. The points above and below the rounding threshold will get different hash codes. (Color figure online)

For the above example, we can also provide a geometric interpretation. In this part, the $\mathbf{b}$ is relaxed as $\mathbf{v} \in \mathbb{R}^K$ in order to describe the classification process in real space. We mainly discuss the $\|\mathbf{w}_1\| = \|\mathbf{w}_2\|$ case as shown in Fig. 2. In this scenario, the classification results will only depend on the angle θ' between $\mathbf{w}$ and $\mathbf{v}$. At the training stage, the original algorithm forces only

one decision boundary ℓ to divide all points in Fig. 2(a), where the point $\mathbf{v}^*$, for example, would be classified to the class 1 correctly because of $\theta'_1 < \theta'_2$, but it would get fault bit codes, which is ascribed to the inconsistency of the classification boundary and the rounding threshold. However, our large-margin algorithm require $m\theta'_1 < \theta'_2$ to make the same decision. Therefore, the point $\mathbf{v}^*$ in Fig. 2(b) hasn't been classified to class 1, which means that this case hasn't been converged. When our large-margin algorithm is converged as shown in Fig. 2(c), we can classify the point $\mathbf{v}^*$ correctly on account of $m\theta'_1 < \theta'_2$, and other points are similar. Furthermore, the wide decision margin in Fig. 2(c) contains the rounding threshold, so the point $\mathbf{v}^*$ will also get true bit codes.

It's obvious that the decision boundary in Fig. 2(a) is turned into a wide decision margin in Fig. 2(c), which both improves the classification generalization and leads to less quantization error. This conclusion will holds for both $\|\mathbf{w}_1\| > \|\mathbf{w}_2\|$ and $\|\mathbf{w}_1\| < \|\mathbf{w}_2\|$ scenarios, too.

Large-Margin Extension: Before further displaying our LMSH algorithm, we should focus on the Eq. (7) again. Equation (7) just works with $0 \leq \theta_1 \leq \frac{\pi}{m}$, while we need a more flexible substitute which works at least with $0 \leq \theta_1 \leq \pi$. Like [15], we replace the $\cos(m\theta)$ with the following formulation:

$$\psi(\theta) = (-1)^k \cos(m\theta) - 2k, \quad \theta \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right], \tag{8}$$

where $k \in [0, m-1]$ and k is an integer. The Eq. (8) is also a monotone decreasing function, and it is less than $\cos(\theta)$ with $0 \leq \theta \leq \pi$ when $m > 1$.

With the large-margin extension, the objective function in (2) is reformulated as:

$$\begin{aligned} \min_{\mathbf{B},\mathbf{W}} \quad & \sum_{i=1}^{N}\sum_{c=1}^{C} \left(Y_{ci} - \|\mathbf{w}_c\|\|\mathbf{b}_i\|\psi(\theta_{ci})\right)^2 + \lambda\|\mathbf{W}\|^2, \\ s.t. \quad & \mathbf{b}_i \in \{-1, 1\}^K, \quad \theta_{ci} \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right], \end{aligned} \tag{9}$$

where Y_{ci} is the $(ci)^{th}$ element of matrix Y, and $Y_{ci} = 1$ if $\mathbf{x}_i$ belongs to class c and 0 otherwise.

Similarly, the optimization problems (3) and (5) can be transformed as below:

$$\min_{\mathbf{W}} \ \mathcal{L}_{\mathcal{W}} = \sum_{i=1}^{N}\sum_{c=1}^{C} \left(\left(Y_{ci} - \|\mathbf{w}_c\|\|\mathbf{b}_i\|\psi(\theta_{ci})\right)^2 + \frac{\lambda}{N}\|\mathbf{w}_c\|^2\right), \tag{10}$$

$$\min_{\mathbf{V}} \ \mathcal{L}_{\mathcal{V}} = \sum_{i=1}^{N}\sum_{c=1}^{C} \left(Y_{ci} - \|\mathbf{w}_c\|\|\mathbf{v}_i\|\psi(\theta'_{ci})\right)^2, \tag{11}$$

where $\mathbf{v}_i$ is the i^{th} column of $\mathbf{V}$ defined in problem (5). θ'_{ci} is the angle between $\mathbf{w}_c$ and $\mathbf{v}_i$.

4 Learning Algorithms

We can optimize our LMSH with typical gradient decent methods.

W-step. Update $\mathbf{W}$ with $\mathbf{B}$ fixed. Unfolding $\cos(m\theta)$ with $\cos(\theta)$ which can be replaced with $\frac{\mathbf{w}^{\mathrm{T}}\mathbf{b}}{\|\mathbf{w}\|\|\mathbf{b}\|}$, and combining Eq. (8), we define $\mathcal{J}_{\mathcal{W}ci}$ as:

$$\begin{aligned}\mathcal{J}_{\mathcal{W}ci} &= \|\mathbf{w}_c\|\|\mathbf{b}_i\|\psi(\theta_{ci}) = (-1)^k \cdot \|\mathbf{w}_c\|\|\mathbf{b}_i\|\cos(m\theta_{ci}) - 2k \cdot \|\mathbf{w}_c\|\|\mathbf{b}_i\| \\ &= (-1)^k \cdot \|\mathbf{w}_c\|\|\mathbf{b}_i\|\Bigg(\mathrm{C}_m^0\big(\frac{\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i}{\|\mathbf{w}_c\|\|\mathbf{b}_i\|}\big)^m - \mathrm{C}_m^2\big(\frac{\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i}{\|\mathbf{w}_c\|\|\mathbf{b}_i\|}\big)^{m-2} \\ &\quad \Big(1 - \big(\frac{\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i}{\|\mathbf{w}_c\|\|\mathbf{b}_i\|}\big)^2\Big) + \cdots\Bigg) - 2k \cdot \|\mathbf{w}_c\|\|\mathbf{b}_i\|,\end{aligned} \tag{12}$$

where $\frac{\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i}{\|\mathbf{w}_c\|\|\mathbf{b}_i\|} \in \Big[\cos(\frac{k\pi}{m}), \cos(\frac{(k+1)\pi}{m})\Big]$, $\mathrm{C}_m^n = \frac{n(n-1)\cdots(n-m+1)}{m(m-1)\cdots 1}$ and k is an integer that belongs to $[0, m-1]$. In fact, the value of k depends on $\frac{\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i}{\|\mathbf{w}_c\|\|\mathbf{b}_i\|}$. And $\frac{\partial \mathcal{J}_{\mathcal{W}ci}}{\partial \mathbf{w}_c}$ can be computed via:

$$\begin{aligned}\frac{\partial \mathcal{J}_{\mathcal{W}ci}}{\partial \mathbf{w}_c} &= (-1)^k \cdot \Bigg(\mathrm{C}_m^0 \frac{m(\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i)^{m-1}\mathbf{b}_i}{(\|\mathbf{w}_c\|\|\mathbf{b}_i\|)^{m-1}} - \mathrm{C}_m^0 \frac{(m-1)(\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i)^m \mathbf{w}_c}{\|\mathbf{w}_c\|^{m+1}\|\mathbf{b}_i\|^{m-1}} \\ &\quad - \mathrm{C}_m^2 \frac{(m-2)(\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i)^{m-3}\mathbf{b}_i}{(\|\mathbf{w}_c\|\|\mathbf{b}_i\|)^{m-3}} + \mathrm{C}_m^2 \frac{(m-3)(\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i)^{m-2}\mathbf{w}_c}{\|\mathbf{w}_c\|^{m-1}\|\mathbf{b}_i\|^{m-3}} \\ &\quad + \mathrm{C}_m^2 \frac{m(\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i)^{m-1}\mathbf{b}_i}{(\|\mathbf{w}_c\|\|\mathbf{b}_i\|)^{m-1}} - \mathrm{C}_m^2 \frac{(m-1)(\mathbf{w}_c^{\mathrm{T}}\mathbf{b}_i)^m \mathbf{w}_c}{\|\mathbf{w}_c\|^{m+1}\|\mathbf{b}_i\|^{m-1}} + \cdots\Bigg) - 2k \cdot \frac{\|\mathbf{b}_i\|\mathbf{w}_c}{\|\mathbf{w}_c\|}.\end{aligned} \tag{13}$$

Substituting Eq. (12) into Eq. (10), we can reformulate $\mathcal{L}_{\mathcal{W}}$. Then $\frac{\partial \mathcal{L}_{\mathcal{W}}}{\partial \mathbf{w}_c}$ can be further computed with:

$$\frac{\partial \mathcal{L}_{\mathcal{W}}}{\partial \mathbf{w}_c} = \sum_{i=1}^{N}\left(-2 \cdot Y_{ci}\frac{\partial \mathcal{J}_{\mathcal{W}ci}}{\partial \mathbf{w}_c} + 2 \cdot \mathcal{J}_{\mathcal{W}ci}\frac{\partial \mathcal{J}_{\mathcal{W}ci}}{\partial \mathbf{w}_c} + 2 \cdot \frac{\lambda}{N}\mathbf{w}_c\right). \tag{14}$$

As a result, we end up with following update rule for $\mathbf{w}_c$:

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) - \alpha_w \cdot \frac{\partial \mathcal{L}_{\mathcal{W}}}{\partial \mathbf{w}_c}. \tag{15}$$

Here we use notation $x(t)$ to denote the value of a parameter x at some iteration t, and α_w is the learning rate. We also update other columns in $\mathbf{W}$ iteratively using the same rule.

B-step. Update $\mathbf{B}$ with $\mathbf{W}$ fixed. According to the Eq. (11), we can obtain $\frac{\partial \mathcal{L}_{\mathcal{V}}}{\partial \mathbf{v}_i}$ using a similar method as the **W-step**. Then we use the following rule for updating $\mathbf{v}_i$:

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) - \alpha_v \cdot \frac{\partial \mathcal{L}_{\mathcal{V}}}{\partial \mathbf{v}_i}, \tag{16}$$

where α_v is the learning rate. We also update other columns in $\mathbf{V}$ iteratively using the same rule. After the $\mathbf{V}$ is learned in each iteration, we can gain $\mathbf{B}$ using some rounding techniques. It is worth mentioning that an important criterion in designing hash functions is that the generated hash codes should take as much information as possible, which implies a balanced hash function that meets $\sum_{i=1}^{N} h(\mathbf{x}_i) = 0$ for each bit [13,14]. As for our problem, the balancing criterion is as follows:

$$B_{ki} = \begin{cases} 1, & V_{ki} > median(V_{k*}) \\ -1, & otherwise \end{cases}, \tag{17}$$

where $k = 1, 2, \cdots, K$ and $i = 1, 2, \cdots, N$. Furthermore, B_{ki} and V_{ki} are the $(ki)^{th}$ elements of $\mathbf{B}$ and $\mathbf{V}$ respectively, and $median(V_{k*})$ denotes the median value of the k^{th} row of $\mathbf{V}$.

Algorithm 1. Large-Margin Supervised Hashing (LMSH)

Input: Labels $\mathbf{Y} \in \mathbb{R}^{C \times N}$ of training data; the code length K; maximum iteration number T; hyper-parameter λ; learning rates α_w and α_v.
Output: Hash codes $\mathbf{B} \in \{-1, 1\}^{K \times N}$.

```
Random initialize B; and initialize W by Eq. (4);
for t = 1 : T do
    W-step: update W using Eq. (15) for each category;
    B-step: obtain initial V by Eq. (6);
            update V using Eq. (16) for each sample;
            generate B by Eq. (17);
end
Return code matrix B.
```

Finally, we conclude our proposed algorithm named *Large-Margin Supervised Hashing* (LMSH) in Algorithm 1. As we can see, Eqs. (4) and (6) should be used for initializations, which will make the algorithm converged with fewer iterations.

Out-of-Sample Extention: Our LMSH is also a two-step method [11]: learning hash codes $\mathbf{B}$ in the first step, and learning hash functions in the second step. To encode a query sample $\mathbf{x}$ by K bits effectively, We choose RBF kernel hash function as below:

$$h(\mathbf{x}) = sgn(\mathbf{P}^{\mathrm{T}}\phi(\mathbf{x})), \tag{18}$$

where $\phi(\mathbf{x})$ is a p-dimensional vector obtained by the RBF kernel mapping: $\phi(\mathbf{x}) = [\exp(-\|\mathbf{x} - \mathbf{x}_{(1)}\|^2/\sigma), \cdots, \exp(-\|\mathbf{x} - \mathbf{x}_{(p)}\|^2/\sigma)]^{\mathrm{T}}$, where $\mathbf{x}_{(1)}, \cdots, \mathbf{x}_{(p)}$ are p anchor samples randomly selected from the training matrix $\mathbf{X}$, and σ is the kernel width. The $\mathbf{P} \in \mathbb{R}^{p \times K}$ is a projection matrix that embeds the $\phi(\mathbf{x})$ into the K-dimensional space. With learned code matrix $\mathbf{B}$, we can approximately obtain $\mathbf{P}$ by the following scheme:

$$\mathbf{P} = (\phi(\mathbf{X})\phi(\mathbf{X})^{\mathrm{T}})^{-1}\phi(\mathbf{X})\mathbf{B}^{\mathrm{T}}, \tag{19}$$

where $\phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \cdots, \phi(\mathbf{x}_N)] \in \mathbb{R}^{p \times N}$. When it comes to the new queries, we can get their hash codes by our hashing function Eq. (18).

Complexity Analysis: The total time of the training stage is $O(TKN \log N)$ with the typical assumptions that $T, K, C, p \ll N$. For a new query $\mathbf{x}$, its time complexity is $O(pM + pK)$.

5 Experiments

5.1 Datasets and Experimental Setups

We evaluate our method on three image databases (VOC2012, CIFAR-10, ImageNet) with semantic labels. VOC2012 [4] consists of 17,125 images associated with 20 classes, with each image containing multiple semantic labels. We represent each instance in this set by a GIST feature vector of 512-dimension. 2000 images therein are sampled for the query set and the remaining are for the training set. CIFAR-10 [1] includes 60,000 images which are manually labeled as 10 classes. Each image is represented with a 320-dimensional GIST vector. In this dataset, 10% of the total are randomly selected as the testing set. ImageNet [3] is an image database organized according to the WordNet hierarchy. We generate 512 GIST features for each images. In the default case, 50 categories in this set are selected randomly, where each category involves 1100 training samples and 100 query images. In above three databases, we treat two images with at least one common category label as neighbors.

In LMSH, we set the maximum iteration number $T = 5$, the smoothing hyper-parameter $\lambda = 1$, the number of anchor points $p = 2000$, and the learning rates $\alpha_w = 1.0e - 8$ and $\alpha_v = 0.5$ on all the listed datasets with different scales. In addition, we set the angular margin parameter $m = 4$ by default.

We compare our LMSH with some state-of-the-art hashing methods which include unsupervised methods: ITQ [7], SGH [8], and supervised methods: KSH [13], SELVE [25], LFH [24], SDH [22], COSDISH [9]. For all the comparison approaches, we use the publicly available MATLAB codes and tune the parameters as suggested in the corresponding papers. Furthermore, we use randomly sampled 2,000 anchor points, in accordance with our LMSH, for SDH.

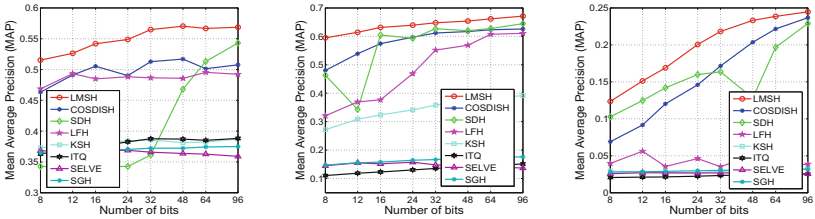
5.2 Results and Analysis

Large Margin or not? To see how much the large-margin extension will contribute to the hash codes learning, we perform a comparison of our methods with or without the large-margin extension. The comparative results, measured by the mean average precision (MAP) [5], are shown in Table 1, where LMSH ($m = 2$) and LMSH ($m = 4$) represents our LMSH methods with the margin parameter set to $m = 2, 4$ respectively, while LMSH ($m = 1$) in fact denotes the parallel method without the large-margin optimization. As we can see, LMSH with $m = 2, 4$ clearly yield more effective hash codes than the one without the large-margin extension ($m = 1$) in three different datasets, which would be due to

the larger angular decision margin that reduces the quantization error. Particularly, the performance gaps between the methods with large-margin optimization or not are increased with longer bits. Besides, compared to LMSH ($m = 2$), LMSH ($m = 4$) still keeps a little advantage in most of scenes, but the gaps are relatively small. This might be because the angular margin with $m = 2$ is large enough, and it's difficult to mine more discriminatory information for a larger angular margin.

Table 1. Comparative mean average precision (MAP) of our methods with different margin values. The best results are in bold, and the second-best ones are underlined.

Method	VOL2012			CIFAR-10			ImageNet (50 categories)		
	32-bits	64-bits	96-bits	32-bits	64-bits	96-bits	32-bits	64-bits	96-bits
LMSH (m = 1)	0.5271	0.5133	0.5037	0.5344	0.4052	0.3897	0.2132	0.2152	0.1901
LMSH (m = 2)	0.5474	0.5454	0.5686	0.6447	0.6588	0.6696	0.2135	**0.2443**	**0.2491**
LMSH (m = 4)	**0.5649**	**0.5668**	**0.5687**	**0.6478**	**0.6618**	**0.6713**	**0.2183**	0.2386	0.2448

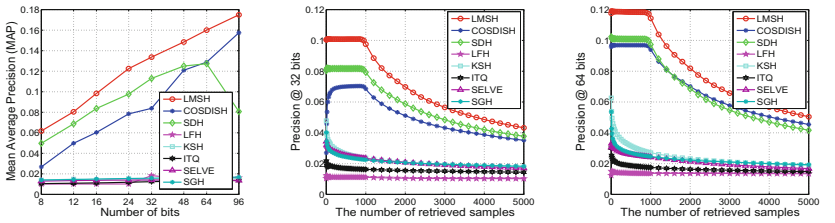


(a) VOC2012 (b) CIFAR-10 (c) ImageNet

Fig. 3. Performance comparison (MAP) of LMSH and other hashing methods.

Results in Three Datasets: We evaluate our method on three image datasets with semantic labels. The Fig. 3(a), (b) and (c) curve the MAP values of all compared methods on VOC2012, CIFAR-10 and ImageNet datasets respectively. As can be seen clearly from Fig. 3, most supervised methods, such as KSH, LFH, SDH and COSDISH, achieve more effective performance than the unsupervised schemes, since the supervised information is involved for training. Thereinto, when hash code is long enough, SDH performs excellent, but it might be lack of stability with shorter bits. Some methods, such as LFH, obtains high performance on the first two datasets, but poorer performances on ImageNet, which probably are ascribed to the complexity of the ImageNet dataset. COSDISH, the up to date method, can acquire a stable and outstanding result, but it's still inferior to our LMSH. Obviously, our LMSH algorithm consistently outperforms all the compared methods in every length of hash code. This is because our large-margin extension significantly encourages inter-class separability and intra-class compactness, which reduce quantization error.

Further Explorations on Large-scale DataSet: To further evaluate the proposed LMSH in a large-scale dataset, we randomly collect more than 100,000 images from 100 different categories in the ImageNet set. Furthermore, 5,000 samples are uniformly selected for the query set. The parameter settings of all hashing methods including our LMSH are identical to the above experiments. The MAP values and the precision curves of topN retrieved images (at 32bits and 64bits) are plotted in Fig. 4. Comparing Figs. 3(c) and 4(a), we can see that the effectiveness of all hashing methods meet a significant decline, probably because more classes lead to a more challenging task. However, the efficacy ranking of all methods varies little. Specifically, LMSH significantly outperforms other algorithms both in MAP and the topN precision varying code length, which exhibits that our proposed approach also have the ability to cope with large-scale datasets.



(a) MAP (b) P v.s. topN (32bits) (c) P v.s. topN (64bits)

Fig. 4. Results on the large scale datasets: ImageNet (100 categories)

6 Conclusion

In this paper, we proposed an novel supervised hashing method with a large angular decision margin whose size can be justified by a preset parameter. The large angular decision margin can encourage the inter-class separability and lead to less quantization error. The experimental results on public datasets demonstrate that our proposed LMSH is an effective and competitive hashing method.

References

1. Alex, K., Hinton, G.: Learning multiple layers of features from tiny images (2009)
2. Deng, C., Liu, X., Mu, Y.: Large-scale multi-task image labeling with adaptive relevance discovery and feature hashing. Sig. Process. **112**, 137–145 (2015)
3. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: ImageNet: a large-scale hierarchical image database. In: CVPR, pp. 248–255 (2009)
4. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes challenge 2012 (VOC2012) results (2012). http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html

5. Lin, G., Shen, C., Shi, Q., Hengel, A., Suter, D.: Fast supervised hashing with decision trees for high-dimensional data. In: CVPR, pp. 1971–1978 (2014)
6. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB, pp. 518–529 (1999)
7. Gong, Y., Lazebnik, S.: Iterative quantization: a procrustean approach to learning binary codes. In: CVPR, pp. 817–824 (2011)
8. Jiang, Q., Li, W.: Scalable graph hashing with feature transformation. In: IJCAI, pp. 2248–2254 (2015)
9. Kang, W., Li, W., Zhou, Z.: Column sampling based discrete supervised hashing. In: AAAI, pp. 1230–1236 (2016)
10. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: NIPS, pp. 1042–1050 (2009)
11. Lin, G., Shen, C., Suter, D., Hengel, A.: A general two-step approach to learning-based hashing. In: ICCV, pp. 2552–2559 (2013)
12. Liu, W., Mu, C., Kumar, S., Chang, S.: Discrete graph hashing. In: NIPS, pp. 3419–3427 (2014)
13. Liu, W., Wang, J., Ji, R., Jiang, Y., Chang, S.: Supervised hashing with kernels. In: CVPR, pp. 2074–2081 (2012)
14. Liu, W., Wang, J., Kumar, S., Chang, S.: Hashing with graphs. In: ICML, pp. 1–8 (2011)
15. Liu, W., Wen, Y., Yu, Z., Yang, M.: Large-margin softmax loss for convolutional neural networks. In: ICML, pp. 507–516 (2016)
16. Liu, X., Fan, X., Deng, C., Li, Z., Su, H., Tao, D.: Multilinear hyperplane hashing. In: CVPR, pp. 5119–5127 (2016)
17. Liu, X., He, J., Deng, C., Lang, B.: Collaborative hashing. In: CVPR, pp. 2147–2154 (2014)
18. Liu, X., He, J., Lang, B., Chang, S.: Hash bit selection: a unified solution for selection problems in hashing. In: CVPR, pp. 1570–1577 (2013)
19. Liu, X., Huang, L., Deng, C., Lu, J., Lang, B.: Multi-view complementary hash tables for nearest neighbor search. In: ICCV, pp. 1107–1115 (2015)
20. Liu, X., Mu, Y., Lang, B., Chang, S.: Compact hashing for mixed image-keyword query over multi-label images. In: ICMR, p. 18 (2012)
21. Liu, X., Mu, Y., Zhang, D., Lang, B., Li, X.: Large-scale unsupervised hashing with shared structure learning. IEEE Trans. Cybern. **45**, 1811–1822 (2015)
22. Shen, F., Shen, C., Liu, W., Shen, H.: Supervised discrete hashing. In: CVPR, pp. 37–45 (2015)
23. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS, pp. 1753–1760 (2008)
24. Zhang, P., Zhang, W., Li, W., Guo, M.: Supervised hashing with latent factor models. In: SIGIR, pp. 173–182 (2014)
25. Zhu, X., Zhang, L., Huang, Z.: A sparse embedding and least variance encoding approach to hashing. IEEE Trans. Image Process. **23**, 3737–3750 (2014)