Multilinear Hyperplane Hashing

Xianglong Liu[†] Xinjie Fan[†] Cheng Deng[‡*] Zhujin Li[†] Hao Su[§] Dacheng Tao[♮]

[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China

[‡]School of Electronic Engineering, Xidian University, Xi'an, China

[§]Department of Computer Science, Stanford University, Stanford, CA, USA

[♮]QCIS and FEIT, University of Technology, Sydney, Australia

xlliu@nlsde.buaa.edu.cn chdeng.xd@gmail.com dacheng.tao@uts.edu.au

Abstract

Hashing has become an increasingly popular technique for fast nearest neighbor search. Despite its successful progress in classic point-to-point search, there are few studies regarding point-to-hyperplane search, which has strong practical capabilities of scaling up applications like active learning with SVMs. Existing hyperplane hashing methods enable the fast search based on randomly generated hash codes, but still suffer from a low collision probability and thus usually require long codes for a satisfying performance. To overcome this problem, this paper proposes a multilinear hyperplane hashing that generates a hash bit using multiple linear projections. Our theoretical analysis shows that with an even number of random linear projections, the multilinear hash function possesses strong locality sensitivity to hyperplane queries. To leverage its sensitivity to the angle distance, we further introduce an angular quantization based learning framework for compact multilinear hashing, which considerably boosts the search performance with less hash bits. Experiments with applications to large-scale (up to one million) active learning on two datasets demonstrate the overall superiority of the proposed approach.

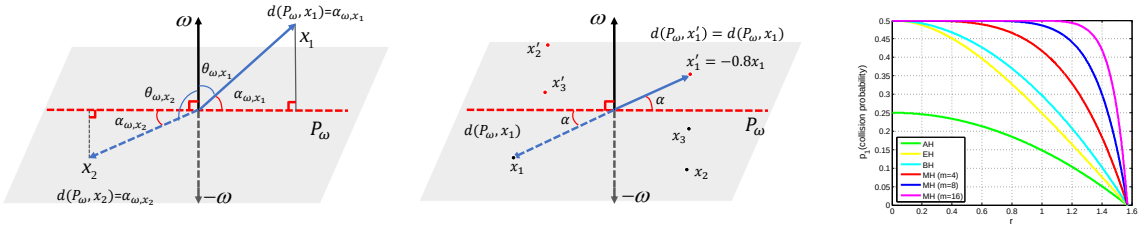
1. Introduction

Recent years have witnessed the success of fast approximated nearest neighbor search in many domains and applications including large-scale visual search [6], object detection [3], classification [8, 15, 21] and recommendation [16, 27]. Locality-sensitive hashing (LSH) [1, 2] pioneered the hash based solution with a good balance between the search performance and computational efficiency. In LSH, the linear projection paradigm was adopted to efficiently generate the hash codes for the specific similarity metric like l_p-norm ($p \in (0, 2]$), and has been widely accepted in many following research for compact hash codes learning [5, 12, 14, 18, 19, 23, 25, 31], equipped with a number of techniques like double bit quantization [9], bit selection [17], asymmetric quantization [24] and discrete optimization [13, 26].

Despite the progress of hashing research, most of existing methods mainly deal with the classic point-to-point nearest neighbor search, where according to certain distance measure the closest database points to the query one are desired as the nearest neighbors. However, in practice there are also a variety of cases that require the nearest points to a hyperplane, namely, point-to-hyperplane search problem. Typical instances includes the large-scale active learning with support vector machines (SVM) [28], maximum margin clustering [32] and large-margin dimensionality reduction [30]. In the task of large-scale active learning with SVMs, the active learning iteration process selects the unlabeled point closest (with minimum-margin) to current SVM's decision hyperplane into the training set, and retrains the SVM classifier with the increasing training data, gaining a provable performance improvement in a few iterations. A common solution for point-to-hyperplane search is the exhaustive search, which nevertheless spends expensive computation and memory on large-scale datasets.

To address this problem, [22] proposed the concomitant hashing to accelerate the min/max inner product, which exploits properties of order statistics of statistically correlated random vectors. Different from the idea of concomitant hashing, prior research [8] first successfully devised hyperplane hashing paradigm with applications to large-scale active learning. Based on the theoretical guarantee of colli-

*corresponding author



(a) point-to-hyperplane search (b) scale-invariance (c) collision probabilities

Figure 1. Illustrations for point-to-hyperplane problem: (a) an simple example, where the point x_2 gives a smaller angle distance $d(\boldsymbol{w}, \boldsymbol{x}_2)$ to the hyperplane $\mathcal{P}(\boldsymbol{w})$ than point x_1; (b) an illustration for the scale-invariance of multilinear hash function with an even order m, where $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ below the hyperplane $\mathcal{P}(\boldsymbol{w})$ respectively share the same angle distance to $\mathcal{P}(\boldsymbol{w})$ and thus same hash code with $\boldsymbol{x}_1'$, $\boldsymbol{x}_2'$ and $\boldsymbol{x}_3'$ locating above $\mathcal{P}(\boldsymbol{w})$; (c) The collision probabilities p_1 of different hyperplane hashing with respect to angle distance r.

sion probability using linear projections, two random hashing methods (AH and EH) have been presented for point-to-hyperplane search with respect to the angle distance and Euclidean distance. The hyperplane hashing permits efficient large-scale search for points near to a hyperplane in sublinear time with a certain accuracy. Following the paradigm based on angle distance, [15] further provided a more strongly locality-sensitive hashing using bilinear hash functions (BH). To meet the same level of search accuracy, BH requires much less hash bits compared with AH and EH. Since the bilinear hash functions are randomly generated in BH, there inevitably exists heavy redundancy among them. Similar to prior hashing research for point-to-point search, a learning based BH (LBH) was designed for compact yet discriminative hash codes, which substantially reduce the storage and computation cost.

As the locality-sensitive property successfully promises fast point-to-hyperplane search, this paper further enhances the property with a much higher collision probability for angle distance. To this end, *we first design a novel hyperplane hashing scheme with multilinear hash functions*, which generates a hash bit based on the product of a series of linear random projections. By devising the coding scheme for both the hyperplane query and the database points, our theoretical analysis shows that with an even number of projections, *the multilinear hash function is more locality sensitive to the angle distance* than AH and BH (BH can be treated as a special case of our multilinear hashing with two projections). This property enables us to achieve satisfying performance using fewer hash bits for practical applications. Besides, *we introduce an angular quantization based learning framework for compact multilinear hyperplane hashing*, better migrating the difficulty of the minimum angle distance search. On two large datasets up to one million, we empirically demonstrate that our approach outperforms state-of-the-arts in active learning with SVMs.

The remaining sections are organized as follows. Section 2 presents the formulation of the point-to-hyperplane search problem. In Section 3 we first review the related hyperplane hashing methods and then introduce the proposed random multilinear hashing with a theoretical analysis. To pursue compact hash codes, we further devise a learning based multilinear hashing algorithm in Section 4. Comprehensive experiments on the task of large-scale active learning over two popular datasets are presented in Section 5, followed by conclusions in Section 6.

2. Problem Formulation

Let us formally state the point-to-hyperplane search problem: given a database $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ of n points in $\mathbb{R}^d$, the goal is to find the closest points in $\mathcal{D}$ to a given hyperplane query $\mathcal{P}_{\boldsymbol{w}}$ determined by the normal vector $\boldsymbol{w} \in \mathbb{R}^d$. Take the margin-based SVM active learning as an example. The margin $\frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b|}{\|\boldsymbol{w}\|}$ of any point $\boldsymbol{x}$ describes the distance to the decision hyperplane $\mathcal{P}_{\boldsymbol{w}}$ of SVM classifier $(\boldsymbol{w}, b)$, and in each iteration the active selection prefers the points with the minimum margin.

Without loss of generality, we append $\boldsymbol{x}$ with a constant 1, and assume that both $\boldsymbol{x}$ and $\boldsymbol{w}$ are normalized and $\mathcal{P}_{\boldsymbol{w}}$ passes through the origin. Then it is easy to see that the point-to-hyperplane nearest neighbor search actually minimizes the absolute value of the cosine between $\boldsymbol{w}$ and $\boldsymbol{x}$:

$$|\cos\theta_{\boldsymbol{w},\boldsymbol{x}}| = \frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}|}{\|\boldsymbol{w}\|\|\boldsymbol{x}\|}. \tag{1}$$

Fig. 1 (a) displays the geometric relationship between the hyperplane normal vector $\boldsymbol{w}$ and database point $\boldsymbol{x}$. Since $|\cos\theta_{\boldsymbol{w},\boldsymbol{x}}| = \sin|\frac{\pi}{2} - \theta_{\boldsymbol{w},\boldsymbol{x}}|$, the distance from $\boldsymbol{x}$ to the hyperplane $\mathcal{P}_{\boldsymbol{w}}$ is monotonically proportional to the angle measure $\alpha_{\boldsymbol{w},\boldsymbol{x}} \in [0, \frac{\pi}{2}]$:

$$\alpha_{\boldsymbol{w},\boldsymbol{x}} = |\frac{\pi}{2} - \theta_{\boldsymbol{w},\boldsymbol{x}}|. \tag{2}$$

Therefore, $\alpha_{\boldsymbol{w},\boldsymbol{x}}$ can serve as the distance metric in our problem: a narrow $\alpha_{\boldsymbol{w},\boldsymbol{x}}$ indicates a close point locating near to the hyperplane $\mathcal{P}_{\boldsymbol{w}}$.

Definition 1 *The distance between a vector $\boldsymbol{x}$ and a hyperplane $\mathcal{P}_{\boldsymbol{w}}$ is measured by $\alpha_{\boldsymbol{w},\boldsymbol{x}}$,* i.e., $d(\mathcal{P}_{\boldsymbol{w}}, \boldsymbol{x}) = \alpha_{\boldsymbol{w},\boldsymbol{x}}$.

Under the angle distance metric, the point-to-hyperplane search problem can equivalently be regarded as a point-to-point search problem between the hyperplane normal vector

Table 1. comparison of collision probability p_1 and hash time t_h of different hash families.

	AH	EH	BH	MH (ours)
p_1	$\frac{1}{4} - \frac{\alpha^2_{\boldsymbol{w},\boldsymbol{x}}}{\pi^2}$	$\frac{\cos^{-1}\sin^2(\alpha_{\boldsymbol{w},\boldsymbol{x}})}{\pi}$	$\frac{1}{2} - \frac{2\alpha^2_{\boldsymbol{w},\boldsymbol{x}}}{\pi^2}$	$\frac{1}{2} - \frac{2^{m-1}\alpha^m_{\boldsymbol{w},\boldsymbol{x}}}{\pi^m}$
t_h	$2kd$	kd^2 or $k\frac{d}{\epsilon^2}$	$2kd$	mkd

$\boldsymbol{w}$ and the database points in $\mathcal{D}$. However, the distance metric is quite different from the conventional nearest neighbor search. Here the nearest neighbors to the hyperplane should be the points almost perpendicular to $\boldsymbol{w}$, rather than those with small angles in point-to-point search. To achieve fast approximate hyperplane query, we will devise a new family of locality sensitive hash function that preserves the specific angle distance with a large probability.

3. Multilinear Hyperplane Hashing

In this section, we will briefly review the existing hyperplane hashing methods, and then propose a random projection based locality sensitive hash of multilinear form with a series of theoretic analysis. Before that, we have to point out that in the whole paper we suppose the binary codes have values from $\{-1, 1\}$ (equivalent to 0/1 code in prior hashing research) for concise derivation.

3.1. Related Work

To our best knowledge, there are only three related hyperplane hash families: angle hyperplane hash (AH), embedding hyperplane hash (EH) [8] and bilinear hash (BH, including its learning based version LBH) [15]. We first give a brief review on these related works.

Angle Hyperplane Hashing: AH as the first hyperplane hashing in the literature employs the two-bit hash function to encode the input $\boldsymbol{x} \in \mathbb{R}^d$:

$$h_{\text{AH}}(\boldsymbol{x}) = \begin{cases} h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{x}, \boldsymbol{x}), & \boldsymbol{x} \text{ is a database point} \\ h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{x}, -\boldsymbol{x}), & \boldsymbol{x} \text{ is a hyperplane normal} \end{cases}$$

where $h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{a}, \boldsymbol{b}) = [\text{sgn}(\boldsymbol{u}^{\text{T}}\boldsymbol{a}), \text{sgn}(\boldsymbol{v}^{\text{T}}\boldsymbol{b})]$, with $\boldsymbol{u}, \boldsymbol{v} \sim \mathcal{N}(0, I_{d\times d})$, i.e., $\boldsymbol{u}$ and $\boldsymbol{v}$ are independent vectors from Gaussian distribution.

Embedding Hyperplane Hashing: Jain et al. also proposed another hyperplane hashing (EH) based on the Euclidean distance [8]. It first computes the high dimensional embedding by vectorizing the rank-1 matrix of the input vector $\boldsymbol{x}$ as $V(\boldsymbol{x}) = vec(\boldsymbol{x}\boldsymbol{x}^{\text{T}}) = (x_1^2, x_1x_2, \cdots, x_1x_d, \cdots, x_d^2)$, and then generates the hash bit using random projection $\boldsymbol{u} \sim \mathcal{N}(0, I_{d^2\times d^2})$

$$h_{\text{EH}}(\boldsymbol{x}) = \begin{cases} \text{sgn}(\boldsymbol{u}^{\text{T}}V(\boldsymbol{x})), & \boldsymbol{x} \text{ is a database point} \\ \text{sgn}(-\boldsymbol{u}^{\text{T}}V(\boldsymbol{x})), & \boldsymbol{x} \text{ is a hyperplane normal} \end{cases}$$

at the cost of expensive computation and storage.

Bilinear Hyperplane Hashing: To suppress the computation cost and simultaneously guarantee high collision probabilities, Liu et al. discovered a new hyperplane hash family named bilinear hashing (BH) [15]. With two projection vectors $\boldsymbol{u}, \boldsymbol{v} \sim \mathcal{N}(0, I_{d\times d})$, BH hash function is defined as:

$$h_{\text{BH}}(\boldsymbol{x}) = \text{sgn}(\boldsymbol{u}^{\text{T}}\boldsymbol{x}\boldsymbol{x}^{\text{T}}\boldsymbol{v}), \tag{3}$$

respectively encoding the database point $\boldsymbol{x}$ and the hyperplane query $\mathcal{P}_{\boldsymbol{w}}$ into $h_{\text{BH}}(\boldsymbol{x})$ and $-h_{\text{BH}}(\boldsymbol{w})$.

Table 1 compares the collision probability and time complexity of different hash families.

3.2. Random Multilinear Hashing

In this paper, we propose a new generic hash family named *multilinear* hash function (MH for short).

Definition 2 *The multilinear hash function $h^m(\cdot) : \mathbb{R}^d \to \{-1, 1\}$ of m-order comprises m linear projection vectors $\boldsymbol{u}_l \sim \mathcal{N}(0, I_{d\times d}), l = 1, \ldots, m$, with the following form*

$$\begin{aligned} h^m(\boldsymbol{x}) &= sgn(\boldsymbol{u}_1{}^{\text{T}}\boldsymbol{x} \cdots \boldsymbol{u}_m{}^{\text{T}}\boldsymbol{x}), \\ i.i.d. \quad \boldsymbol{u}_l &\sim \mathcal{N}(0, I_{d\times d}),\ l = 1, \ldots, m. \end{aligned} \tag{4}$$

Note that following the above definition, when $m = 2$, the multilinear hash function degenerates to BH [15]. The multilinear form enjoys several advantages in hyperplane hashing. First, it helps us focus on the angle distance in our hyperplane hashing, eliminating the effect of the data scale (even the negative one, see Fig. 1(b) for an illustration with an even m). Second, since the projection on a vector can preserve the data locality to some extent, multiple projections together can significantly boost the probability collision. We will show its locality sensitive property in the following theoretical analysis.

Lemma 1 *Given a query point $\boldsymbol{w} \in \mathbb{R}^d$ and a database point $\boldsymbol{x} \in \mathbb{R}^d$, the probability of collision for these two points under h^m is*

$$\mathbf{P}[h^m(\boldsymbol{w}) = h^m(\boldsymbol{x})] = \frac{(1 - \frac{2\theta_{\boldsymbol{w},\boldsymbol{x}}}{\pi})^m + 1}{2}$$

Proof (1) When $m = 1$, the equation holds according to the fact from [1].

(2) When $m > 1$, assuming the equation holds for all $k < m$, we have,

$$\begin{aligned} &\mathbf{P}[h^m(\boldsymbol{w}) = h^m(\boldsymbol{x})] \\ &= \mathbf{P}[h^{m-1}(\boldsymbol{w}) = h^{m-1}(\boldsymbol{x})]\mathbf{P}[\text{sgn}(\boldsymbol{u}_m{}^{\text{T}}\boldsymbol{w}) = \text{sgn}(\boldsymbol{u}_m{}^{\text{T}}\boldsymbol{x})] \\ &\quad + \mathbf{P}[h^{m-1}(\boldsymbol{w}) \neq h^{m-1}(\boldsymbol{x})]\mathbf{P}[\text{sgn}(\boldsymbol{u}_m{}^{\text{T}}\boldsymbol{w}) \neq \text{sgn}(\boldsymbol{u}_m{}^{\text{T}}\boldsymbol{x})] \\ &= \mathbf{P}[h^{m-1}(\boldsymbol{w}) = h^{m-1}(\boldsymbol{x})](1 - \frac{\theta_{\boldsymbol{w},\boldsymbol{x}}}{\pi}) \\ &\quad + (1 - \mathbf{P}[h^{m-1}(\boldsymbol{w}) = h^{m-1}(\boldsymbol{x})])\frac{\theta_{\boldsymbol{w},\boldsymbol{x}}}{\pi} \\ &= \frac{(1 - \frac{2\theta_{\boldsymbol{w},\boldsymbol{x}}}{\pi})^m + 1}{2}. \end{aligned}$$

This completes the proof. □

Lemma 2 *Given a hyperplane query $\mathcal{P}_{\boldsymbol{w}}$ with the normal vector $\boldsymbol{w} \in \mathbb{R}^d$, define $h^m(\mathcal{P}_{\boldsymbol{w}}) = -h^m(\boldsymbol{w})$. Then, the probability of collision for $\mathcal{P}_{\boldsymbol{w}}$ and $\boldsymbol{x}$ under the random multilinear hash h^m with an even m is*

$$\mathbf{P}[h^m(\mathcal{P}_{\boldsymbol{w}}) = h^m(\boldsymbol{x})] = \frac{1}{2} - \frac{2^{m-1}\alpha_{\boldsymbol{w},\boldsymbol{x}}^m}{\pi^m}$$

Proof Using $h^m(\mathcal{P}_{\boldsymbol{w}}) = -h^m(\boldsymbol{w})$ for $\mathcal{P}$, we get

$$\begin{aligned}\mathbf{P}[h^m(\mathcal{P}_{\boldsymbol{w}}) = h^m(\boldsymbol{x})] &= 1 - \mathbf{P}[h^m(\boldsymbol{w}) = h^m(\boldsymbol{x})] \\ &= 1 - \frac{(1 - 2\frac{\theta_{\boldsymbol{w},\boldsymbol{x}}}{\pi})^m + 1}{2} = \frac{1}{2} - \frac{2^{m-1}(\frac{\pi}{2} - \theta_{\boldsymbol{w},\boldsymbol{x}})^m}{\pi^m}.\end{aligned}$$

Then by applying the facts $\alpha_{\boldsymbol{w},\boldsymbol{x}} = |\theta_{\boldsymbol{w},\boldsymbol{x}} - \frac{\pi}{2}|$ and m is even, one can complete the proof. □

Lemma 2 gives us several hints to the design of multilinear hyperplane hash functions. First, m should be even, otherwise, one can find that the collision probability of $\mathcal{P}_{\boldsymbol{w}}$ and $\boldsymbol{x}$ under h^m reaches the highest when $\theta_{\boldsymbol{w},\boldsymbol{x}}$ equals π. That is to say it is still very likely that the collision happens even if $\alpha_{\boldsymbol{w},\boldsymbol{x}}$ equals to $\frac{\pi}{2}$, which contradicts to our intention. Second, under the mild condition that m is even, the collision probability can be amplified considerably by increasing the order m of the multilinear function. Fig. 1(b) demonstrates the necessity of an even m for the angle distance preservation, possessing the invariance to the magnitude and reverse of the point vector.

Theorem 1 *Under the condition that m is even, the multilinear hyperplane hash function family h^m is $\left(r, r(1+\epsilon), \frac{1}{2} - \frac{2^{m-1}r^m}{\pi^m}, \frac{1}{2} - \frac{2^{m-1}(r(1+\epsilon))^m}{\pi^m}\right)$-sensitive to the distance measure $d(\mathcal{P}_{\boldsymbol{w}}, \boldsymbol{x}) = \alpha_{\boldsymbol{w},\boldsymbol{x}}$ with $r,\ \epsilon > 0$.*

Proof Using Lemma 2, if $d(\boldsymbol{x}, \mathcal{P}_{\boldsymbol{w}}) \leq r$, we have

$$\begin{aligned}\mathbf{P}[h^m(\mathcal{P}_{\boldsymbol{w}}) = h^m(\boldsymbol{x})] &= \frac{1}{2} - \frac{2^{m-1}d^m(\mathcal{P}_{\boldsymbol{w}}, \boldsymbol{x})}{\pi^m} \\ &\geq \frac{1}{2} - \frac{2^{m-1}r^m}{\pi^m} = p_1.\end{aligned}$$

Likewise, when $d(\mathcal{P}_{\boldsymbol{w}}, \boldsymbol{x}) > r(1+\epsilon)$ we have

$$\mathbf{P}[h^m(\mathcal{P}_{\boldsymbol{w}}) = h^m(\boldsymbol{x})] < \frac{1}{2} - \frac{2^{m-1}(r(1+\epsilon))^m}{\pi^m} = p_2,$$

and $p_1 > p_2$. This completes the proof. □

Theorem 1 indicates that the locality sensitivity to the angle distance is bounded by the collision probability which monotonically increases with respect to even m, and for any even $m > 2$ this probability is larger than that of AH, EH and BH (see Fig. 1(c) and Table 1). Note that using the similar multilinear trick, we can also improve the collision probability of AH and EH theoretically. Meanwhile, we can see that there is a tradeoff between the high collision probability and low time complexity for bit generation. Next, we give the theoretical performance guarantee and computation bound when using multilinear hash functions for point-to-hyperplane nearest neighbor search.

Theorem 2 *Given a database $\mathcal{D}$ with n points and a hyperplane query $\mathcal{P}_{\boldsymbol{w}}$, if there exists a database point $\boldsymbol{x}^*$ such that $d(\boldsymbol{x}^*, \mathcal{P}_{\boldsymbol{w}}) \leq r$, then with $\rho = \frac{\ln p_1}{\ln p_2}$ (1) using n^ρ hash tables with $\log_{1/p_2} n$ hash bits, the random multilinear hyperplane hash of an even order is able to return a database point $\hat{\boldsymbol{x}}$ such that $d(\hat{\boldsymbol{x}}, \mathcal{P}_{\boldsymbol{w}}) \leq r(1+\epsilon)$ with probability at least $1 - \frac{1}{c} - \frac{1}{e}$, $c \geq 2$; (2) the query time is sublinear to the entire data number n, with $n^\rho \log_{1/p_2} n$ bit generations and cn^ρ pairwise distances computation.*

The theorem guarantees the practicality of MH that the nearest neighbors can be located fast (in a sublinear time) with a large probability. The proof can be easily completed based on the locality sensitivity. Please refer to the proofs in the supplementary material.

4. Learning Multilinear Hash Functions

Though randomized hashing methods enjoy elegant theoretic guarantee and computational simplicity, they usually require long hash codes for satisfying performance, and consequently need much computation and memory cost in practice. Conventional hashing research for point-to-point search problem attempted to address such issue by learning hash functions from data [12, 14, 31], which can generate compact hash codes by eliminating the redundancy among functions, and thus largely save computation and storage [2]. Following this line, in this section we introduce an angle quantization based learning method for the multilinear hyperplane hash functions (LMH for short).

4.1. Angle Quantization

Our motivation comes from the fact that hyperplane-to-point search problem is tightly connected to the point-to-point one. In detail, as shown in Fig. 1 (a) and guaranteed in Lemma 2, a hyperplane query $\mathcal{P}_{\boldsymbol{\omega}}$ can be equivalently represented by its normal $\boldsymbol{\omega}$ with the multilinear hash bit $h_j^m(\mathcal{P}_{\boldsymbol{w}}) = -h_j^m(\boldsymbol{w})$ generated by the j-th multilinear function $h_j^m(\boldsymbol{w}) = \text{sgn}((\boldsymbol{u}_1^j)^{\mathrm{T}}\boldsymbol{w} \cdots (\boldsymbol{u}_m^j)^{\mathrm{T}}\boldsymbol{w})$.

If we denote the k-length multilinear hash code of $\boldsymbol{x}$ by

$$\boldsymbol{H}^m(\boldsymbol{x}) = (h_1^m(\boldsymbol{x}), \cdots, h_k^m(\boldsymbol{x}))^{\mathrm{T}}, \tag{5}$$

then $\boldsymbol{H}^m(\mathcal{P}_{\boldsymbol{w}}) = -\boldsymbol{H}^m(\boldsymbol{w})$, and searching the nearest neighbors to $\mathcal{P}_{\boldsymbol{\omega}}$ turns to finding those points $\boldsymbol{x}$ having the most similar hash codes (or smallest Hamming distances) to $\boldsymbol{H}^m(\mathcal{P}_{\boldsymbol{w}})$. Namely, for desired points $\boldsymbol{H}^m(\mathcal{P}_{\boldsymbol{w}})^{\mathrm{T}}\boldsymbol{H}^m(\boldsymbol{x}) \to k$ with a small angle distance $\alpha_{\boldsymbol{w},\boldsymbol{x}}$.

Therefore, preserving the angle plays the most important role in the discriminative hash function learning.

Since the hash codes actually serve as a binary representation for points in a more complex feature space [4, 5, 7], in our problem the learnt binary codes should maximally preserve the angle (or cosine similarity) among the points. To this end, we propose angle quantization based learning algorithm, which aims at capturing the angles among the projected database points using the binary hash codes. Instead of random generation, here the projection vectors of each multilinear hash function will be learnt to minimize the angle quantization loss, and thus those points close to the hyperplane query can be discriminatively distinguished and located quickly in the Hamming space.

Formally, given a training set $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n'})$, our goal is to preserve the angles among them by learning k multilinear hash functions of m order, characterized by m projection matrice $\boldsymbol{U}_l = (\boldsymbol{u}_l^1, \cdots, \boldsymbol{u}_l^k)$, $l = 1, \ldots, m$. Denote the projection values before binary quantization by

$$\boldsymbol{Y} = \boldsymbol{U}_1{}^{\mathrm{T}}\boldsymbol{X} \odot \cdots \odot \boldsymbol{U}_m{}^{\mathrm{T}}\boldsymbol{X},$$

where the symbol $\odot$ represents the Hadamard product, then the binary codes will be $\boldsymbol{B} = \mathrm{sgn}(\boldsymbol{Y}) = \boldsymbol{H}^m(\boldsymbol{X})$, and the projection and the binary code of each $\boldsymbol{x}_i$ will be the corresponding columns of $\boldsymbol{Y}$ and $\boldsymbol{B}$, denoted by $\boldsymbol{Y}_i$ and $\boldsymbol{B}_i$.

Therefore, for each point $\boldsymbol{x}_i$ its hash code $\boldsymbol{B}_i$ should maximally approximate $\boldsymbol{Y}_i$. The learning problem can be formulated as follows:

$$\begin{aligned} \max_{\boldsymbol{B},\boldsymbol{U}_l} \quad & \sum_{i=1}^{n'} \cos(\boldsymbol{B}_i, \boldsymbol{Y}_i) \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{1} = 0, \quad \boldsymbol{U}_l^{\mathrm{T}}\boldsymbol{U}_l = \boldsymbol{I}, \quad l = 1, \ldots, m. \end{aligned} \tag{6}$$

The constraint $\boldsymbol{U}_l^{\mathrm{T}}\boldsymbol{U}_l = \boldsymbol{I}$ makes sure the k hash functions are independent to each other, while $\boldsymbol{B}\boldsymbol{1} = \boldsymbol{0}$ forces the data to be evenly distributed over -1 and 1 for a balanced coding, and we relax it to $\boldsymbol{Y}\boldsymbol{1} = \boldsymbol{0}$ due to the optimization difficulty stemming from the discrete constraint of $\boldsymbol{B}$.

4.2. Iterative Optimization

Note that in problem (6), if we denote the projection of all training data using j-th hash function by $\boldsymbol{y}_j = (Y_{j1}, \cdots, Y_{jn'})^{\mathrm{T}} \in \mathbb{R}^{n'\times 1}$, and the binary codes by $\boldsymbol{b}_j = (B_{j1}, \cdots, B_{jn'})^{\mathrm{T}} \in \{-1, 1\}^{n'\times 1}$, then the problem can be approximately reformulated as:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n'} \cos(\boldsymbol{B}_i, \boldsymbol{Y}_i) \approx \sum_{i=1}^{n'} \frac{\boldsymbol{B}_i^{\mathrm{T}}}{\sqrt{k}} \boldsymbol{Y}_i \\ & = \frac{1}{\sqrt{k}} \sum_{i=1}^{n'} \sum_{j=1}^{k} B_{ji} Y_{ji} = \frac{1}{\sqrt{k}} \sum_{j=1}^{k} \boldsymbol{b}_j{}^{\mathrm{T}} \boldsymbol{y}_j \end{aligned}$$

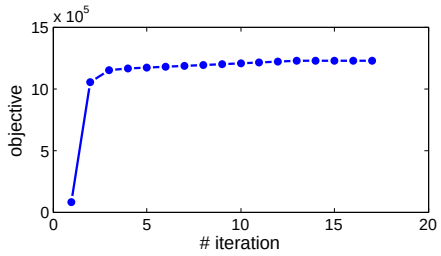


Figure 2. The objective with respect to the iteration number.

This rearrangement of the formula inspires our sequential optimization solution: one by one we learn $\boldsymbol{u}_l^j$, $1 \le l \le m$ and the bit values $\boldsymbol{b}_j$ for all training data. At each iterative step, $\boldsymbol{b}_j$ and $\boldsymbol{u}_l^j$ are optimized in an alternating manner by fixing others. Specifically at the j-th step, we repeat the following updating until the objective converges:

(Step 1) Update $\boldsymbol{b}_j$. When $\boldsymbol{u}_1^j, \cdots, \boldsymbol{u}_m^j$ are fixed, $\boldsymbol{y}_j = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{u}_1^j \odot \cdots \odot \boldsymbol{X}^{\mathrm{T}}\boldsymbol{u}_m^j$ is determined, we solve $\boldsymbol{b}_j$ by

$$\begin{aligned} \max_{\boldsymbol{b}_j} \quad & \boldsymbol{b}_j^{\mathrm{T}}\boldsymbol{y}_j \\ \text{s.t.} \quad & \boldsymbol{b}_j \in \{-1, 1\}^{n'\times 1} \end{aligned} \tag{7}$$

Clearly, the optimal solution is given by $\boldsymbol{b}_j = \mathrm{sgn}(\boldsymbol{y}_j)$.

(Step 2) Update $\boldsymbol{u}_l^j$, $l = 1, \ldots, m$. This can be efficiently done by alternatingly optimizing each $\boldsymbol{u}_l^j$ while fixing $\boldsymbol{u}_{l'}^j$ for any $l' \neq l$. Since all $\boldsymbol{u}_l^{j'}$, $j' < j$ have been learnt in previous steps, by introducing the vector

$$\boldsymbol{e} = \left(\odot_{l'\neq l} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{u}_{l'}^j\right), \tag{8}$$

the optimization problem with respect to $\boldsymbol{u}_l^j$ turns to be

$$\begin{aligned} \max_{\boldsymbol{u}_l^j} \quad & \boldsymbol{a}^{\mathrm{T}}\boldsymbol{u}_l^j \\ \text{s.t.} \quad & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{u}_l^j = 0, \ (\boldsymbol{u}_l^{j'})^{\mathrm{T}}\boldsymbol{u}_l^j = 0, \ j' < j, \end{aligned} \tag{9}$$

where $\boldsymbol{a} = \boldsymbol{X}(\boldsymbol{e} \odot \boldsymbol{b}_j)$ and $\boldsymbol{c} = \boldsymbol{X}\boldsymbol{e}$ corresponds to the relaxed balanced constraint $\boldsymbol{1}\boldsymbol{y}_j = 0$, and $(\boldsymbol{u}_l^{j'})^{\mathrm{T}}\boldsymbol{u}_l^j = 0$ is imposed to enforce the diversity of the projections.

This is a standard linear programming problem, whose optimal solution can be efficiently obtained using a number of optimizing techniques like primal-dual interior point [20]. In the above alternating optimization, we get a suboptimal solution in each update, and with a few iterations we can easily obtain the optimized multilinear hash functions h_j^m in the j-th step.

To start the learning, we randomly initialize the projection vectors of each hash function, and then learn a series of multilinear functions that boost the accuracy of the point-to-hyperplane search. The whole learning based multilinear hash method is listed in Algorithm 1. Fig. 2 shows the objective curves with respect to the number of iterations,

Algorithm 1 Learning Multilinear Hyperplane Hash.

Input: training data $\boldsymbol{X}$, the order of the multilinear hash m and the code length k;
Initialize: the projection matrice $\boldsymbol{U}_l = (\boldsymbol{u}_l^1, \cdots, \boldsymbol{u}_l^k)$, $\boldsymbol{u}_l \sim \mathcal{N}(0, I_{d\times d}), l = 1, \ldots, m$;
for $j = 1, \ldots, k$ **do**
  repeat
    Update each binary codes $\boldsymbol{b}_j$ by solving (7);
    Update each projection matrice $\boldsymbol{u}_l^j$ by solving (9), $l = 1, \ldots, m$;
  until Converge
end for
Output: the hashing projection vectors $\boldsymbol{U}_l$, $l = 1, \ldots, m$ and the binry codes $\boldsymbol{B}$.

where we can observe that our algorithm can converge fast, and thus the hash functions can be learnt efficiently at the training stage.

Note that in LMH we mainly devote our efforts to preserving the angle values instead of the absolute ones. Nevertheless, the scale invariance of the multilinear hash makes us only have to focus on the half of the subspace partitioned by the hyperplane query, and a bit flipping over the query code can further improve the recall performance in practice. Though the learnt MH can hardly be proved locality sensitive, exploiting the data structures faithfully helps us pursue more discriminative functions than random way. Our experiments in the next section will demonstrate this point.

5. Experiments

The studied point-to-hyperplane search problem appears in many tasks like classification with active learning and cutting-plane based maximum margin clustering. In our experiments, we adopt the widely-studied active learning with SVMs as the application, and comprehensively evaluate the proposed MH and LMH in terms of search accuracy and efficiency. Besides, four state-of-the-art hyperplane hashing algorithms AH, EH, BH and its learning version LBH, and two naive but common methods including exhaustive linear scan and random selection are also evaluated.

5.1. Datasets and Settings

Experiments are conducted on two large widely-adopted datasets in classification including **MNIST** [11] and **Tiny-1M** appended with **CIFAR-10** [10]. **MNIST** is a dataset of handwritten digits, comprising 60,000 training images and 10,000 test images associated with digits from 0 and 9 (*i.e.*, 10 different classes). Each image is represented by a 784-dimensional vector corresponding to its 28×28 grayscale pixel intensities. **Tiny-1M** comprises two parts from 80M tiny image set [29]: one million images randomly sampled without labels and **CIFAR-10** with 60,000 32×32 color images associated with 10 semantic categories, each of which has 6,000 samples. Following the setting in [15], we treat the first part as the "other" class, and represent each image using the provided 384-dimensional GIST descriptors.

In the SVM based classification with active learning, we separately train linear SVMs in the one-vs-all setting for each class with random initializations, *i.e.*, 5 labeled samples for each class on both datasets. Then different point-to-hyperplane methods are adopted to select the unlabeled samples nearest to the current SVM's hyperplane. In all experiments we run 300 active selection iterations, and retrain SVMs using the training set appended with the selected samples in each iteration.

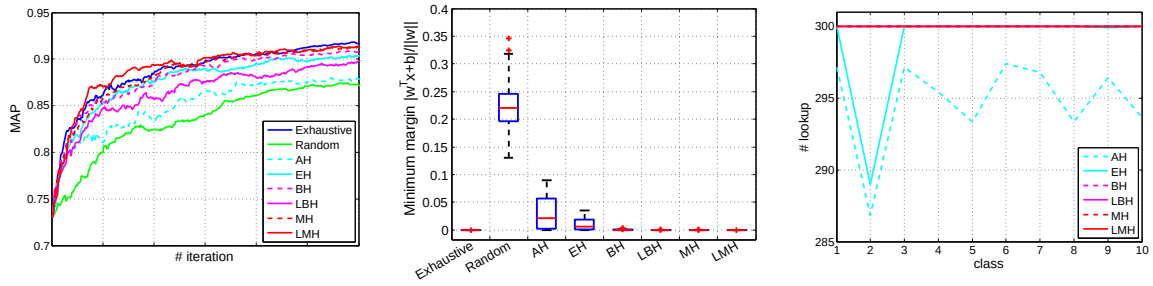
5.2. Results and Discussions

We compare the proposed MH and the learn based version LMH to two naive methods: exhaustive and random selection, and four state-of-the-art hyperplane hashing methods: AH, EH [8], BH, and LBH [15].

At the offline stage, we first generate hash functions for each hashing methods, and encode the database points using the same number of hash bits for fair comparison. According to the size of each set, we respectively generate 16 hash bits for hash tables on **MNIST** and 20 bits on **Tiny-1M**. This is because the theoretical and practical analysis show that the optimal code length should be close to $\log_2 n$ [7]. When it comes to the online point-to-hyperplane search in each active learning step, from each table we lookup the buckets with hash codes in a small Hamming radius from the query code, and then obtain the nearest point to the hyperplane by ranking candidates in the buckets. To balance the efficiency and accuracy, in our experiments we empirically set the radius to 5 for both datasets. For the case that no points fall within the radius, we alternatively choose the random selection as a supplement.

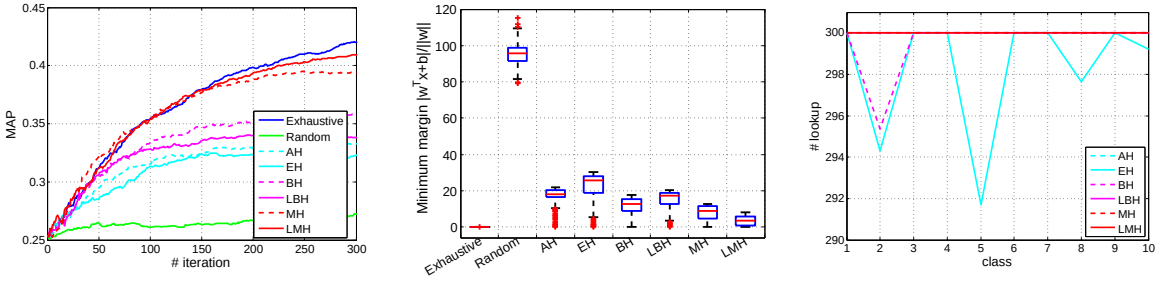
5.2.1 Performance in Large-Scale Active Learning

We first report the results using different methods on MNIST. Fig. 3(a) plots the corresponding mean average precision (MAP) curves with respect to 300 active learning iterations. Clearly, BH and LBH outperform AH, EH and random selection, and our MH and LMH ($m = 4$) further improve the MAP and consistently obtains the best performances among all hyperplane hashing methods, mainly owning to the fact that the multilinear functions preserve the discriminative power of the angle distance, and thus retrieve the most perpendicular points to the hyperplane normal. Moreover, we can notice that at the first few iterations, LMH even gets a better performance than the exhaustive way, which indicates that there is no guarantee that the exhaustive selection can serve as the best choice for better



(a) active learning curves (b) minimum-margins (c) # success table lookups

Figure 3. Results of the active learning using different methods on MNIST.



(a) active learning curves (b) minimum-margins (c) # success table lookups

Figure 4. Results of the active learning using different methods on Tiny-1M.

test accuracy. In all iterations, LMH and MH consistently achieve very close performance to the exhaustive selection. This is mainly because MH can faithfully find the nearest neighbors for each hyperplane query. The minimum margins shown in Fig. 3(b) also illustrate this point, where LMH and MH get very small margins with a slight variation.

Fig. 3(c) further depicts the number of nonempty table lookup per class, accumulated in the 300 iterations. It can be observed that in certain cases AH and EH fail to retrieve candidate points for minimum-margin based active selection. Instead, owning to the higher collision probability, our MH and LMH can enjoy the 100% success rate for hash table lookup given the specific Hamming radius.

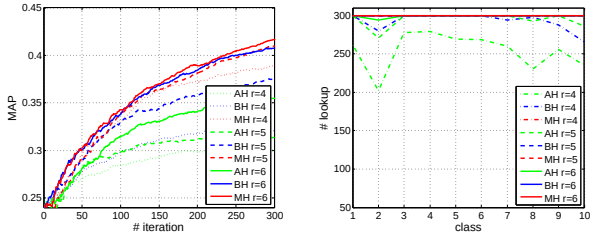
To demonstrate the practicability of our hyperplane hashing method for large-scale point-to-hyperplane search problem, we perform active selection on the Tiny-1M dataset with 1.06 millon images. We use the provided 10K images from CIFAR-10 as the testing samples, and the rest 1.01 millon as the database for active sample selection. Fig. 4 shows the similar results on the Tiny-1M in terms of MAP, minimum margins and success lookup number. Namely, as the exhaustive selection does, both of our hashing methods ($m = 4$) select the points with smaller margins than other hashing baselines (Fig. 4(b)), and hit the desired points with a higher success rate (Fig. 4(c)). Subsequently, they can obtain satisfying performances for the practical applications, and our LMH can further learn more compact yet discriminative codes by exploiting the angle information among data, and again obtains the best performance.

5.2.2 Performance using Different Settings

In above experiments, we adopt one hash table and a relatively large lookup radius to achieve the desired recall performance, which grantees that the nearest neighbor to the hyperplane query can be located. To comprehensively study the performance of MH with different settings, we further vary the parameters including the lookup radius r, the table number L and the order m of the multilinear hash function.

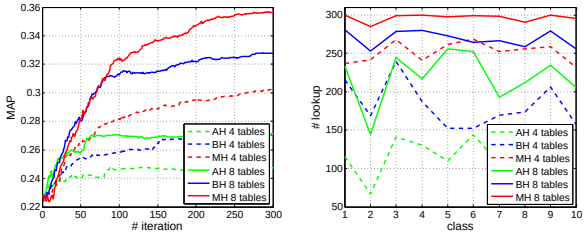
Lookup radius: Fig. 5 investigates the precision and the recall performance in terms of the active learning curves and the number of the success lookups. Here we vary the lookup radius r from 4 to 6 in one 20-bits hash table and treat all points belonging to the buckets within r as the ranking candidates. It can be observed that for each method, as the radius increases, higher precision and success lookups can be attained. This is because that the enlarged search range guarantees a high probability that the true nearest neighbors can be identified among the retrieved candidates. Besides, we can see that in all cases our MH can get the best performance. For instance, MH using a mall lookup radius $r = 4$ can even get a much better performance than the baselines using a large one, *i.e.*, BH $r = 5$ and AH using $r = 6$.

Table number: In practice, either adopting a large lookup radius or building multiple hash tables can significantly increase the recall for nearest neighbor search. Therefore, in Fig. 6 we further study the performance of different hyperplane hash methods with more than one (4 and 8) hash tables. In order to clearly observe the effect when using more hash tables, in this experiment we set the lookup radius to a



(a) active learning curves (b) # success table lookups

Figure 5. Results of the active learning using different methods with different lookup radius on Tiny-1M.



(a) active learning curves (b) # success table lookups

Figure 6. Results of the active learning using different methods with a different number of tables on Tiny-1M.

small one (*i.e.*, $r = 2$). In this case, the ranking candidates are those points in each table that fall in the buckets within r Hamming distance to the hyperplane query. According to Fig. 6(a), it is obvious that more hash tables can largely boost the point-to-hyperplane search for different methods, mainly owing to the improved recall of the nearest neighbors. This conclusion can also be verified by the results of success table lookups in Fig. 6(b). Moreover, in all cases our MH again obtains the best performance, with the encouraging locality sensitivity.

Multilinear order: The above experimental results have demonstrated that MH can achieve satisfying performance with a high collision probability. Our theoretical analysis implies that a large m will further increase such probability, enhancing the locality sensitive property. To study its effect in the application of active learning, Table 2 lists the MAP and minimum margins of the 300th active learning iteration using $m = 4, 8, 16$. Consistent with Theorem 1, using more projection in one hash function will amplify the collision probability of the informative samples, and thus increase the precision of SVM classifiers, meanwhile decreasing the minimum margins.

Computational efficiency: The computational efficiency is regarded as a critical issue in large-scale applications. Among all methods, only LBH and LMH require offline training, where the iterative optimization in LMH is faster than Nesterov's gradient method in LBH [15]. As to the search time, Fig. 7 shows the averaged time (in seconds) of the active selection using different methods on Tiny-1M. Exhaustive selection requires linearly scanning all the points in the unlabeled database and thus suffers from much more time consumption than others, while random selection serves as the most fast method, but wastes most labeling time on those uninformative samples. Utilizing the table indexing, the hashing methods can avoid the expensive computation by the sublinear search and ranking on a quite small candidate set. Due to the dimension expansion, the code generation in EH takes more computation and memory than others. Although our hyperplane hashing methods depend on more linear projections than AH and BH (*i.e.*, the hashing time t_h linearly increases with m), but it is ignorable ($\leq 10^{-3}$s when $m \leq 16$) compared to other parts of the active selection time, *i.e.*, ranking time. From the figure, we can observe that its total selection time varies very slightly, and is quite close to baseline hashing methods. Therefore, it can be concluded that MH can give a sufficiently satisfying performance without consuming too much computation.

Table 2. Classification accuracy and minimum margins of the 300th iteration using MH with different m on Tiny-1M.

	$m = 4$	$m = 8$	$m = 16$
MAP	$38.82_{\pm 1.03}$	$40.32_{\pm 0.81}$	$41.67_{\pm 0.41}$
$\min \frac{\lvert \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}+b \rvert}{\lVert \boldsymbol{w} \rVert}$	$7.13_{\pm 2.11}$	$6.32_{\pm 1.77}$	$4.41_{\pm 0.83}$

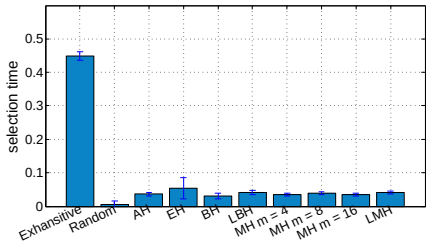


Figure 7. Selection time (seconds) of active learning using MH with different m on Tiny-1M.

6. Conclusions

In this paper, we considered the point-to-hyperplane search problem and proposed a new family of hyperplane hash function named multilinear hashing. The proposed multilinear hash function enjoys strong locality sensitivity for the angle distance, and thus is able to retrieve the nearest points for a hyperplane query in a sublinear time. By learning the projections from the data, we can further preserve the angles among data using compact yet discriminative hash codes, which largely enables the practicability of the multilinear hyperplane hashing in many applications. Large-scale active learning experiments on two datasets have demonstrated the superior performance of the multilinear hashing in terms of both accuracy and efficiency.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (61402026 and 61572388), and Australian Research Council Projects DP-140102164, FT-130101457, and LE140100061.

References

[1] M. Charikar. Similarity estimation techniques from rounding algorithms. *In STOC*, pages 380–388, 2002. 1, 3

[2] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *ACM SOCG*, pages 253–262, 2004. 1, 4

[3] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *IEEE CVPR*, pages 1–8, 2013. 1

[4] Y. Gong, S. Kumar, V. Verma, and S. Lazebnik. Angular quantization-based binary codes for fast similarity search. In *NIPS*, pages 1196–1204, 2012. 5

[5] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE CVPR*, pages 817 –824, 2011. 1, 5

[6] J. He, J. Feng, X. Liu, T. Cheng, T.-H. Lin, H. Chung, and S.-F. Chang. Mobile product search with bag of hash bits and boundary reranking. In *IEEE CVPR*, pages 3005–3012, 2012. 1

[7] J. He, S. Kumar, and S.-F. Chang. On the difficulty of nearest neighbor search. In *ICML*, pages 1–8, 2012. 5, 6

[8] P. Jain, S. Vijayanarasimhan, and K. Grauman. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. In *NIPS*, pages 928–936. 2010. 1, 3, 6

[9] W. Kong and W.-J. Li. Double-bit quantization for hashing. In *AAAI*, pages 2604–2623, 2012. 1

[10] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009. 6

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. 6

[12] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. Dick. Learning hash functions using column generation. In *ICML*, pages 1–8, 2013. 1, 4

[13] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014. 1

[14] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011. 1, 4

[15] W. Liu, J. Wang, Y. Mu, S. Kumar, and S.-F. Chang. Compact hyperplane hashing with bilinear functions. In *ICML*, pages 1–8, 2012. 1, 2, 3, 6, 8

[16] X. Liu, J. He, C. Deng, and B. Lang. Collaborative hashing. In *IEEE CVPR*, pages 2147–2154, 2014. 1

[17] X. Liu, J. He, and B. Lang. Reciprocal Hash Tables for Nearest Neighbor Search. In *AAAI*, pages 626–632, 2013. 1

[18] X. Liu, L. Huang, C. Deng, J. Lu, and B. Lang. Multi-view complementary hash tables for nearest neighbor search. In *IEEE ICCV*, pages 1107–1115, 2015. 1

[19] X. Liu, Y. Mu, B. Lang, and S.-F. Chang. Compact hashing for mixed image-keyword query over multi-label images. In *ACM ICMR*, page 18, 2012. 1

[20] S. Mehrotra. On the implementation of a primal-dual interior point method. 1992. 5

[21] Y. Mu, G. Hua, W. Fan, and S.-F. Chang. Hash-svm: Scalable kernel machines for large-scale visual classification. In *IEEE CVPR*, pages 979–986, 2014. 1

[22] Y. Mu, J. Wright, and S.-F. Chang. Accelerated large scale optimization by concomitant hashing. In *ECCV*, pages 414–427, 2012. 1

[23] Y. Mu and S. Yan. Non-metric locality-sensitive hashing. In M. Fox and D. Poole, editors, *AAAI*, pages 539–544, 2010. 1

[24] B. Neyshabur, N. Srebro, R. Salakhutdinov, Y. Makarychev, and P. Yadollahpour. The power of asymmetry in binary hashing. In *NIPS*, pages 2823–2831, 2013. 1

[25] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011. 1

[26] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *IEEE CVPR*, pages 37–45, 2015. 1

[27] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *In NIPS*, pages 1–8, 2008. 1

[28] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, Mar. 2002. 1

[29] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE TPAMI*, 30(11):1958–1970, Nov 2008. 6

[30] C. Xu, D. Tao, C. Xu, and Y. Rui. Large-margin weakly supervised dimensionality reduction. In *ICML*, pages 865–873, 2014. 1

[31] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *ICML*, pages 1–8, 2014. 1, 4

[32] B. Zhao, F. Wang, and C. Zhang. Efficient maximum margin clustering via cutting plane algorithm. In *SDM*, pages 751–762, 2008. 1