Ordinal Constrained Binary Code Learning for Nearest Neighbor Search

Hong Liu,[†‡] **Rongrong Ji,**[†‡*] **Yongjian Wu,**[♮] **Feiyue Huang**[♮]
[†]Fujian Key Laboratory of Sensing and Computing for Smart City, Xiamen University, 361005, China
[‡]School of Information Science and Engineering, Xiamen University, 361005, China
[♮]BestImage Lab, Tencent Technology (Shanghai) Co.,Ltd, China
lynnliu.xmu@gmail.com, rrji@xmu.edu.cn, littlekenwu@tencent.com, garyhuang@tencent.com

Abstract

Recent years have witnessed extensive attention in binary code learning, *a.k.a.* hashing, for nearest neighbor search problems. It has been seen that high-dimensional data points can be quantized into binary codes to give an efficient similarity approximation via Hamming distance. Among existing schemes, ranking-based hashing is recent promising that targets at preserving ordinal relations of ranking in the Hamming space to minimize retrieval loss. However, the size of the ranking tuples, which shows the ordinal relations, is quadratic or cubic to the size of training samples. By given a large-scale training data set, it is very expensive to embed such ranking tuples in binary code learning. Besides, it remains a dificulty to build ranking tuples efficiently for most ranking-preserving hashing, which are deployed over an ordinal graph-based setting. To handle these problems, we propose a novel ranking-preserving hashing method, dubbed *Ordinal Constraint Hashing* (OCH), which efficiently learns the optimal hashing functions with a graph-based approximation to embed the ordinal relations. The core idea is to reduce the size of ordinal graph with ordinal constraint projection, which preserves the ordinal relations through a small data set (such as clusters or random samples). In particular, to learn such hash functions effectively, we further relax the discrete constraints and design a specific stochastic gradient decent algorithm for optimization. Experimental results on three large-scale visual search benchmark datasets, *i.e.* LabelMe, Tiny100K and GIST1M, show that the proposed OCH method can achieve superior performance over the state-of-the-arts approaches.

Introduction

Learning binary code, *a.k.a.* hashing, to preserve the data similarity has recently been popular in various computer vision and artificial intelligence applications, *e.g.*, image retrieval (Liu et al. 2016), objective detection (Dean et al. 2013), multi-task learning (Weinberger et al. 2009), linear classifier training (Li et al. 2011; Lin et al. 2014), and active learning (Liu et al. 2012b). In this setting, real-valued data points are encoded into binary codes that are significantly efficient in storage and computation. In general, most hashing methods learn a set of hash functions $h^k : \mathbf{R}^d \rightarrow \{0,1\}^r$. It typically maps the d-dimensional data space into an r-bit discrete Hamming space, such that the nearest neighbors can be approximated by using the compact binary codes learned.

Recent advances in binary code learning can be categorized into either data-independent or data-dependent ones (Wang et al. 2016). The former typically refers to random projection/partition of feature space, such as Locality Sensitive Hashing (LSH) and Min-Hash (MinHash). It typically requires long bits or multi-hash table to achieve satisfied retrieval performance. Both supervised and unsupervised hashing belong to data-dependent hashing. Unsupervised hashing, learns hash functions by preserving the data structure, distribution, or topological information, *e.g.*, Spectral Hashing (SH) (Weiss, Torralba, and Fergus 2008), Anchor Graph Hashing (AGH) (Liu et al. 2011), Isotropic Hashing (IsoHash) (Kong and Li 2012), Iterative Quantization (ITQ) (Gong et al. 2013), Discrete Graph Hashing (DGH) (Liu et al. 2014), Spherical Hashing (SpH) (Heo et al. 2015), Scalable Graph Hashing (SGH) (Jiang and Li 2015), and Ordinal Embedding Hashing (OEH) (Liu et al. 2016). Differently, supervised hashing aims to learn more accurate hash functions with label information. Representative works include, but not limited to, Binary Reconstructive Embedding (BRE) (Kulis and Darrell 2009), Minimal Loss Hashing (MLH) (Norouzi and Fleet 2011), Kernel-based Supervised Hashing (KSH) (Liu et al. 2012a), Semi-Supervised Hashing (SSH) (Wang, Kumar, and Chang 2012), Supervised Discrete Hashing (SDH) (Shen et al. 2015).

Although promising performance has been shown from these methods, we argue that, the relative order among data must be preserved in the Hamming space rather than pairwise relations. So, many ranking-based hashing algorithms have been proposed to learn more discriminative hash codes, *e.g.*, Hamming Distance Metric Learning (HDML) (Norouzi, Fleet, and Salakhutdinov 2012), Ranking-based Supervised Hashing (RSH) (Wang et al. 2013a), Struct-based Hashing (StructHash) (Lin, Shen, and Wu 2014), Top-Rank Supervised Binary Coding (Top-RSBC) (Song et al. 2015). However, most of these methods adopt the stochastic gradient decreasing (SGD) optimization under triplet ordinal constraints, which needs massive iterations. On the other

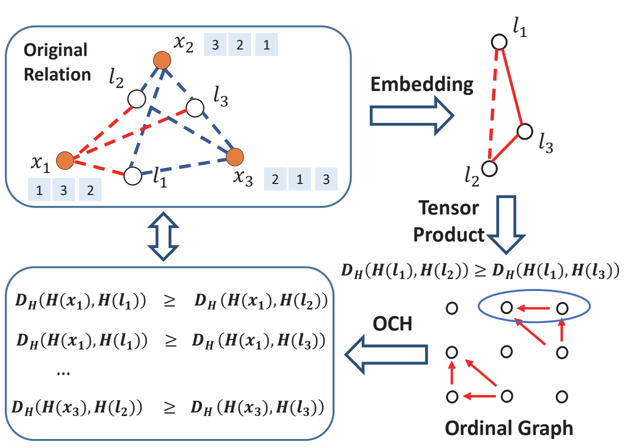

Original Relation
x_2 3 2 1
l_2
l_3
x_1
l_1
x_3
1 3 2
2 1 3
Embedding
l_1
l_3
l_2
Tensor Product
$D_H(H(l_1),H(l_2)) \geq D_H(H(l_1),H(l_3))$
$D_H(H(x_1),H(l_1)) \geq D_H(H(x_1),H(l_2))$
$D_H(H(x_1),H(l_1)) \geq D_H(H(x_1),H(l_3))$
...
$D_H(H(x_3),H(l_2)) \geq D_H(H(x_3),H(l_3))$
OCH
Ordinal Graph

Figure 1: The Framework of our proposed Ordinal Constraint Hashing (OCH).

hand, such ranking-based hashing are categorized into supervised hashing, which is typically labor-intensive to obtain sufficient semantic labels in many real-world applications.

In this paper, we mainly focus on ranking preserved hashing, termed Ordinal Constraint Hashing (OCH), with two key innovations to address the issues raised above. Firstly, OCH attempts to preserve the ordinal relation from the high-dimensional feature space to the Hamming space. Our learning procedure takes into account the *quartic ordinal relations*, rather than the pairwise or triplet ordinal relations that are widely used in the earlier works. To make use of the quartic relations, the ranking lists for each individual queries are converted to a quartic tuples representation that can be presented as Eq. 1 below. Formally, let x be the data point, a quartic tuples can be given as:

$$\mathbb{J}=\{(q;x_i,x_j,x_k)|D(q,x_i)<D(q,x_j)<D(q,x_k)\}, \quad (1)$$

where $D(\cdot,\cdot)$ is the distance measure (*e.g.* Euclidean distance). To the best of our knowledge, none existing hashing methods, even ranking-based hashing (Li et al. 2013; Wang et al. 2013b; Liu et al. 2016), have considered such ordinal relation among data points, which can be obtained in a large-scale manner with extensive human labor.

Inspired by the recent work (Babenko, Arandjelović, and Lempitsky 2016), OCH embeds in which the original quartic order relation can hold as the triplet order relation. And the size of order tuples can be reduced from the original $[n^4]$ to $[L^3]$ ($n << L$) to avoid high complexity computation cost, where n is the number of training data and L is the number of the sub-set (*e.g.* clusters). Then, we propose a general method to construct the ordinal graph with tensor product calculation. Such construction scheme avoids the time consuming selection way to represent the ordinal tuples. The OCH minimizes the inconsistency between the given ordinal relation tuples and the ones derived from the corresponding hash codes. At last, due to the constraint of orthogonal projection, the hash functions are learned via a special SGD algorithm, which formulate the problem as combing the traditional SGD with Stiefel manifold optimization. The whole framework is shown in Fig. 1. We compare the proposed OCH against various state-of-the-art unsupervised hashing methods on three widely used similarity search benchmarks, *i.e.*, **LabelMe**, **Tiny100K**, and **GIST1M**. Quantitative experiments demonstrate that OCH outperforms the existing unsupervised hashing methods in terms of both accuracy and efficiency.

The rest of this paper is organized as follows: In Section 2, we briefly overview the related works of the proposed method. Section 3 and 4 describe the proposed OCH and the iterative SGD based optimization. In Section 5, we show and analyze the experimental results. Finally, we conclude this paper in Section 6.

Background and Related Work

We briefly introduce the problem of binary code learning and review related work as below:

Binary Code Learning aims to learn a set of hash functions to encode real-valued feature points to compact binary codes. For a data point $x \in \mathbb{R}^d$, the hash functions $H = \{h_1, ..., h_r\}$ produces a r-bit binary code $y = \{y_1, y_2, ..., y_r\}$ for x as:

$$y = [h_1(x), ..., h_k(x), ..., h_r(x)]. \quad (2)$$

Therefore, the k-th hash bit y_k is calculated by $h_k(x) = sgn\big(f_k(x)\big)$, where $sgn(\cdot)$ is the sign function that returns 1 if $f_k(x) > 0$ and -1 otherwise. Such hash functions are encoded as a mapping process combining with quantization, which has been widely used in many traditional hashing algorithm, *e.g.*, LSH. And the function $f_k : \mathbb{R}^d \rightarrow \mathbb{R}$ is a linear transformation, given by $f_k(x) = W_k^T x + b$ with the projection matrix W_k and an offset $b \in \mathbb{R}$. Then given a set of hash functions, the database $X \in \mathbb{R}^{d\times n}$ with n samples are mapped to the produced binary codes as

$$Y = \big\{h_1(X), ..., h_k(X), ..., h_r(X)\big\}, \quad (3)$$

where $Y \in \{0, 1\}^{r\times n}$ is the hash code matrix of the database X.

Ordinal Embedding Hashing (Liu et al. 2016) aims to learn a set of hash functions as Eq. 3. Such functions can preserve the ordinal relations between δ_{ij} and δ_{kl}, where δ_{ij} is the dissimilarity between the i-th and the j-th item. Its goal is to make sure the ordinal relation can be preserved in the produced Hamming space: $\delta_{ij} < \delta_{kl} : \|H(x_i) - H(x_j)\|_1 < \|H(x_k) - H(x_l)\|_1$. To embed such ordinal relations, OEH first constructs a directed unweighted ordinal graph $G = (V, E) = [n^4]$, where each node v_{ij} is the dissimilar degree δ_{ij}, and each directed edge is defined via $e_{(i,j,k,l)} = (v_{ij} \rightarrow v_{kl}) \subseteq E$. Then the objective function is to minimize the inconsistency between the given ordinal relation graph and the ones generated from the corresponding hash codes. At last, by using the landmark-based ordinal graph, the quartic ordinal relation is transformed to the triplet ordinal relation, which transforms the target of OEH to $\delta_{ij} < \delta_{ik} : \|H(x_i) - H(l_j)\|_1 < \|H(x_i) - H(l_k)\|_1$ where l_j and l_k are the landmark points.

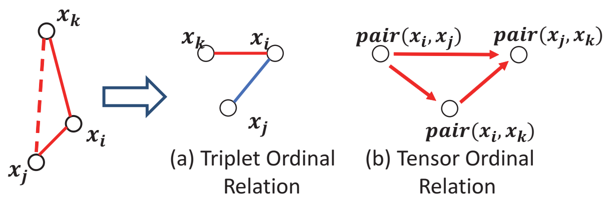


Figure 2: The ordinal relation for a training data.

Ordinal Constraint Hashing

In this section, we describe the proposed OCH in detail. Let $X = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^{d\times n}$ be the data matrix with n samples, where x_i is the i-th column of X with d dimensions.[1] As the same definition as δ_{ij} before, we suppose that $\delta_{ii} = 0$ and $\delta_{ij} = \delta_{ji}$, the comparison $\delta_{ij} < \delta_{kl}$ reflects the data pair (i, j) is more similar than pair (k, l). Then an oridinal relation set C can be given as: $\{\delta_{ij} < \delta_{ik} < \delta_{il} \mid \forall(i; j, k, l) \in C\}$. We further define a K-means centers points matrix $L = \{l_1, l_2, ..., l_L\} \in \mathbb{R}^{d\times L}$, where $L << n$.

The proposed OCH aims to learn the hash functions by embedding the ordinal relation. To this end, a straightforward method is to maximize the loss function between the ordinal relation set C and the corresponding relation in the Hamming space. This can be defined by the following objective function:

$$\begin{aligned} max \sum_{(i;j,k,l)\in C} & I\big(D_H(b_i, b_j) \le D_H(b_i, b_k) \le D_H(b_i, b_l)\big) \\ s.t.\ \ b_i = sgn(W^T x_i),&\ \ W^T W = I,\ \ W \in \mathbb{R}^{d\times r}, \end{aligned} \tag{4}$$

where $I(\cdot)$ is an indicator function which returns 1 if the condition is satisfied and 0 otherwise, and $D_H(b_i, b_j)$ returns the Hamming distance between hash code b_i and b_j.

The first key problem of OCH is how to represent the ordinal relation set directly. In the previous works, the ordinal relation is represented with the triplet data (x_i, x_j, x_k), where the pair (x_i, x_j) is formed by two nearest neighbors under Euclidean distance, and (x_i, x_k) is dissimilar pair. This ordinal relation is shown in Fig. 2 (a), where red line represent the dissimilar pair and blue line represent the similar pair. However, this triplet representation needs to randomly select the corresponding triplet tuples and compare all the data points side by side, which time consuming and memory costly. Even, it is hard to define the similar and dissimilar data pairs for unsupervised learning problems.

In this paper, we address the above issue by proposing a novel ordinal graph model with tensor product calculation, in which triplet or even quartic ordinal relation can be efficiently represented. Given a dataset X and the similarity measure (*e.g.* Euclidean Distance), the affinity graph $S \in \mathbb{R}^{n\times n}$ is constructed as follows:

$$S(i, j) = \begin{cases} 0, & i = j, \\ e^{-\|x_i - x_j\|_2^2/2\sigma^2}, & otherwise, \end{cases} \tag{5}$$

[1]Without loss of generality, assume that the data X is normalized and mean-centered.

where $S(i, j)$ represents the similarity between two data points. We further define a dissimilarity graph $DS \in \mathbb{R}^{n\times n}$, with each entry as $DS(i, j) = 1/S(i, j)$ and $DS(i, i) = 0$ (which is the definition of δ_{ij}).

Then, a tensor ordinal graph (TOG) $\mathbf{G}$ is defined as:

$$\mathbf{G} = S \otimes DS, \tag{6}$$

where $\otimes$ is the Kronecker product of matrices defined as $\mathbf{G}(ij, kl) = S(i, j) \cdot DS(k, l)$. Thus, each entry of $\mathbf{G}$ relates to four data points. When S and DS are two $n \times n$ matrices, $\mathbf{G}$ is an $n^2 \times n^2$ matrix. Therefore, the ordinal relation between the quartic items in $(i, j, k, l) \in C$ can be represented through the TOG, as following:

$$\begin{cases} \delta_{ij} < \delta_{kl}, & \mathbf{G}(ij, kl) > 1, \\ \delta_{ij} > \delta_{kl}, & \mathbf{G}(ij, kl) \le 1. \end{cases} \tag{7}$$

Fig.2 (b) shows a toy example. Under such a circumstance, the ordinal relation in original feature space is $\delta_{ij} < \delta_{ik} < \delta_{jk}$, which can be simply calculated and compared with Euclidean distance. And for the quartic item (i, k, i, j), the (ik, ij)-th entry is $G(ik, ij) = S(i, k) \cdot DS(i, j) = S(i, k)/S(i, j)$. Due to the relation of $S(i, k) < S(i, j)$, we can get $\mathbf{G}(ik, ij) < 1$, which reflects the truth ordinal relation $\delta_{ik} > \delta_{ij}$ in the original space. In such way, the proposed TOG can represent the ordinal relation simply with tensor product scheme.

Although the TOG approximates the relation set C easily, it is not expect that the larger size of TOG makes the calculation time consuming. To solve this problem, according to the relation in set C, we transform the constraint $\{\delta_{ij} < \delta_{ik} < \delta_{il} \mid \forall(i; j, k, l) \in C\}$ as follows:

$$\begin{aligned} \mathbf{O} = \sum_{\forall(i\neq j,k,l)} I\big((\|x_i - x_j\|_2^2 - \|x_i - x_k\|_2^2)^2 \\ - (\|x_i - x_j\|_2^2 - \|x_i - x_l\|_2^2)^2\big). \end{aligned} \tag{8}$$

That is to say, minimizing Eq. 8 is to approximate the ordinal constraint set C over all dataset. Since the points in the dataset have been normalized, we rewrite Eq. 8 as:

$$\begin{aligned} \mathbf{O} &= \sum_{\forall(i\neq j,k,l)} I\big((x_i^T x_j - x_i^T x_k)^2 - (x_i^T x_j - x_i^T x_l)^2\big) \\ &= \sum_{\forall(j,k,l)} I\big((x_j - x_l)^T M (x_j - x_l) \\ &\qquad - (x_j - x_k)^T M (x_j - x_k)\big), \end{aligned} \tag{9}$$

where $M = \sum_i x_i^T x_i$ is a positive semi-definite symmetrical matrix. So it is convenient to use SVD to decompose it into $Z \in \mathbb{R}^{d_{svd}\times d}$ such that $M = Z^T \Lambda Z$. Then a mapping function can be defined as $u_i = Z x_i \in \mathbb{R}^{d_{svd}}$, and Eq. 9 is written as following:

$$\begin{aligned} \mathbf{O} &= \sum_{\forall(j,k,l)} I\big((u_j - u_l)^T \Lambda (u_j - u_l) \\ &\qquad - (u_j - u_k)^T \Lambda (u_j - u_k)\big) \\ &\le \sum_{\forall(j,k,l)} I\big(\|\Lambda^{\frac{1}{2}}\|_2^2 \cdot (\|u_j - u_l\|_2^2 - \|u_j - u_k\|_2^2)\big) \\ &\propto \sum_{\forall(j,k,l)} I\big(\|u_j - u_l\|_2^2 - \|u_j - u_k\|_2^2\big). \end{aligned} \tag{10}$$

By means of this mapping, termed ordinal constraint projection (OCP), we have projected the original ordinal relation to an approximation ordinal relation set $\{\hat{\delta}_{ij} < \hat{\delta}_{ik} \mid \forall(i,j,i,k) \in \hat{C}\}$, which can be generated from the TOG easily.

On the other hand, the total number of ordinal constraints is still too large to be used for training. Inspired by the Product Quantization (Jegou, Douze, and Schmid 2011), the distance $D(u_i, u_j)$ can be approximated by the distance $D(u_i, u_j) \approx D(a_i, a_j)$, where $a_i = Zl_i$ is the embedding center point. In our setting, we further approximate the original ordinal set C by a sub-set of ordinal relation set after the aforementioned OCP. Then the size of the tensor ordinal graph can be reduced to $[L^4]$ $(L << n)$, where L is the number of K-means centers. From Eq. 10, we can use the triplet relation among centers for the original quartic relation approximation, which significantly reduce the scale of ordinal graph from $[n^4]$ to $[L^3]$.

Therefore, the overall objective function in Eq. 4 for the proposed OCH approach is rewritted as follows:

$$\begin{aligned} &min \sum_{(i,j,i,k)\in\hat{C}} I\big(D_H(b_i, b_j) \geq D_H(b_i, b_k)\big) \\ &s.t.\, b_i = sgn(V^T a_i),\ \ VV^T = I,\ \ V \in \mathbb{R}^{d_{svd}\times r}. \end{aligned} \tag{11}$$

Due to $W^T W = I$ with $W = Z^T V$, we can easily get the new orthogonal constraint in Eq. 11, which needs to be hold during optimization. Meanwhile our target is to learning the hash functions that hold the ordinal relations in the Hamming space. As a result, a new optimization scheme should be designed, which is introduced subsequently in Section 4.

Optimization

Directly minimizing the objective function in Eq. 11 is intractable, as the coding function is discrete while the Hamming space is not continuous. To solve this problem, we relax the discrete constraints from the Hamming space to an approximated continuous space.

To that effect, we first relax the hashing function $H(a_i) = sgn(V^T a_i)$ as follows:

$$\hat{H}(a_i) = tanh(V^T a_i), \tag{12}$$

where $tanh(\cdot)$ is a good approximation for $sign(\cdot)$ that transforms the binary codes from $\{0, 1\}$ to $\{-1, 1\}$. Correspondingly, the Hamming distance is calculated as:

$$D_H(b_i, b_j) = \frac{1}{2}\big(r - \hat{H}^T(a_i) \cdot \hat{H}(a_j)\big). \tag{13}$$

Finally, we use the Sigmoid function to replace the indicator function for convenient optimization and avoid overfitting. Based upon the above relaxations, the objective function in Eq. 11 can be rewritten as:

$$F(V|G) = \sum_{(i,j,i,k)\in\hat{C}} p(i,j,i,k),\ \ s.t.\ VV^T = I, \tag{14}$$

where the $p(\cdot,\cdot,\cdot,\cdot)$ is the Sigmoid function defined as follows:

$$p(i,j,i,k) = \frac{1}{1 + exp\big(D_H(b_i, b_k) - D_H(b_i, b_j)\big)}.$$

Algorithm 1 Ordinal Embedding Hashing (OEH)

```
Input: Data set $X = \{x_1, x_2, ..., x_n\}$, parameters $\gamma$ and $\eta$.
Output: The hash function $H(x_i) = sgn(V^T Z x_i)$.
1: Generate centers $L$ by K-means algorithm;
2: Generate matrix $Z$ and embed $Z$ into the $L$;
3: Generate ordinal relations set $\hat{C}$ by Tensor Ordinal
   Graph in Eq. 7;
4: repeat
5:     Randomly select a subset $c$ from set $\hat{C}$;
6:     Calculate the gradient according to Eq. 16;
7:     Update $V$ according to Eq. 15;
8: until convergence or reaching the maximum iteration
   number.
```

Intuitively, the gradient descent approach can be used to carry out an iterative optimization for Eq. 14. However, due to the orthogonal constraints of projection matrix V, the objective function is non-convex, which is also hard to optimize. In the following, we further introduce an alternative stochastic gradient descent algorithm on Stiefel Manifold to solve this problem efficiently.

Stochastic Gradient Descent on Stiefel Manifold

Optimization with respect to the orthogonal constraints has been recently studied in (Wen and Yin 2013; Ge et al. 2014). In particular, to solve Eq. 14, most straightforward method uses gradient descent on the Stiefel manifold defined by $\mathcal{O} = \{V \in \mathbb{R}^{d_{svd}\times r}, VV^T = I\}$ (Absil, Mahony, and Sepulchre 2007). An off-the-shelf iterative solver has been developed in (Wen and Yin 2013), which is however sensitive to the initialization and hard to integrated into our optimization.

Recent advances in (Cunningham and Ghahramani 2015) are to calculated the objective F, gradients ∇F together in the full space $\mathbb{R}^{d_{svd}\times r}$. These gradients are then projected into a tangent space $\mathbb{T}$ with the transformation $\mathbf{P} : \mathbb{R} \to \mathbb{T}$, and a retraction $\mathbf{R} : \mathbb{T} \to \mathcal{O}$ is adopted to map the gradients from tangent space to the targeted Stiefel manifold space[2]. Therefore, this generic algorithm offers a global convergence proof for such a method by a line search (Absil, Mahony, and Sepulchre 2007). Finally, the updating rule of the optimal projection matrix can be defined as:

$$V = \mathbf{R}\big(\eta \mathbf{P}(-\nabla F)\big) \tag{15}$$

where η is the choice of convergence parameter, which is widely used for first-order optimization. In this way, the gradient of Eq. 14 in the full space is given by:

$$\nabla F = \sum_{c\subset\hat{C}_s} \big(p(c)(1-p(c))\big) \cdot \left[\frac{\partial D_H(b_i, b_k)}{\partial V} - \frac{\partial D_H(b_i, b_j)}{\partial V}\right], \tag{16}$$

where I is an identity matrix, c is a subset random selected

[2]Both two transformation functions have been defined in the appendix of (Cunningham and Ghahramani 2015).

Table 1: The *m*AP and Precision Comparison Using Hamming Ranking on Two Benchmark with Different Hash Bits

Methods	LabelMe						Tiny100K					
	mAP			Pre@100			mAP			Pre@100		
	32	64	128	32	64	128	32	64	128	32	64	128
LSH	0.1582	0.2563	0.3555	0.2894	0.4233	0.5543	0.1231	0.1831	0.2323	0.2524	0.3845	0.4626
AGH	0.2099	0.2349	0.2297	0.3895	0.4424	0.4730	0.1176	0.1350	0.1221	0.4286	0.3751	0.3687
IsoHash	0.2606	0.3225	0.3883	0.4486	0.5211	0.5939	0.1864	0.2295	0.2599	0.4107	0.4906	0.5064
SpH	0.2481	0.3096	0.3859	0.4476	0.5244	0.6232	0.1931	0.2541	0.3193	0.4404	0.5599	**0.6755**
SGH	0.3012	0.3849	0.4477	0.4899	0.5920	0.6589	0.2054	0.2637	0.3038	0.4444	0.5380	0.5969
ITQ	0.3059	0.3753	0.4210	0.5012	0.5724	0.6176	0.2039	0.2436	0.2612	0.3044	0.4912	0.5164
OEH	0.2111	0.3546	0.4449	0.3693	0.5642	0.6500	0.1650	0.2495	0.3086	0.3635	0.4986	0.5954
OCH	**0.3140**	**0.3947**	**0.4620**	**0.5072**	**0.6028**	**0.6713**	**0.2297**	**0.2919**	**0.3379**	**0.4805**	**0.5785**	0.6314

from the whole ordinal relations $\hat{C}$, and the gradient of Hamming distance is formulated as:

$$\frac{\partial D_H(b_i, b_j)}{\partial V} = -\frac{1}{2}\{a_i \cdot \left[\left(1\text{-}\hat{H}^2(a_i)\right) \odot \hat{H}(a_j)\right]^T + a_j \cdot \left[\left(1\text{-}\hat{H}^2(a_j)\right) \odot \hat{H}(a_i)\right]^T\}. \tag{17}$$

In Eq. (17), $\odot$ is the Hadamard product which represents the element-wise product.

The details of the proposed SGD on Stiefel manifold is shown in Algorithm 1. The overall training complexity of the proposed algorithm is $O(trL^3 d_{svd} + nL)$, where t is the number of iterations. It is linear to the training set and related to the complexity of the K-means step, which is faster than the previous work (Liu et al. 2016) in ranking preserved hashing. The experiments shown in the next section prove that the proposed OCH has superior performance for large-scale similarity retrieval with high efficiency in training.

Experiments

In this section, we evaluate the proposed Ordinal Constraint Hashing on three large-scale benchmarks, *i.e.*, LableMe, Tiny100K, and GIST1M, which are widely used for evaluating nearest neighbor search algorithms.

Datasets

We briefly summarize the datasets used as below: The **LabelMe** dataset consists of $22,019$ images, each of which is represented by a 512-dimensional GIST feature (Oliva and Torralba 2001). The **Tiny-100K-384D** dataset consists of $100K$ images sampled from the TinyImages dataset (Torralba, Fergus, and Weiss 2008), each of which is represented by a 384-dimensional GIST descriptors. The **GIST-1M-960D** dataset is introduced in (Jegou, Douze, and Schmid 2011), which consists of one million images described by GIST descriptors.

Baseline Methods

We compared the proposed OCH with several representative and state-of-the-art unsupervised hashing methods, including Local Sensitive Hashing (**LSH**) (Datar et al. 2004), Anchor Graph Hashing (**AGH**) (Liu et al. 2011), Isotropic Hashing (**IsoHash**) (Kong and Li 2012), Iterative Quantization (**ITQ**) (Gong et al. 2013), Spherical Hashing (**SpH**) (Heo et al. 2015), Scalable Graph Hashing (**SGH**) (Jiang and Li 2015), and Ordinal Embedding Hashing (**OEH**) (Liu et al. 2016).[3] For all the compared methods, we carefully follow the original parameter setting in respective datasets. We implement our OCH hashing using MATLAB on a single PC with Duo-Core I7-3421 and 75G memory, where the complete data set can be stored.

Evaluation Protocols

To evaluate the proposed hashing algorithm, we adopt a set of widely used protocols in recent papers (Jiang and Li 2015; Park, Cafarella, and Mozafari 2015; Liu et al. 2016). For a given query, the top 2% ranking items with Euclidean distances are defined as with the same label of the query. Then, based on the Euclidean ground-truth, we compute the recall curve, precision@100 ranking curve, and mean average precision. For all the experiments, $2,000$ data points are randomly selected as test set, and the remaining are used as the dataset. To avoid the overfitting, we randomly select $10,000$ points as the training set for all the algorithms, which has been used in (He, Wen, and Sun 2013). We run all the experiments 10 times and report the average performance.

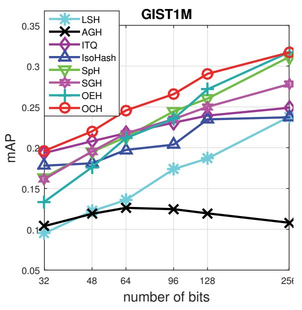
Parameter Tunning

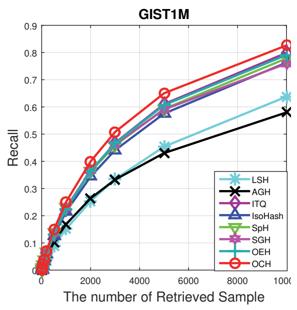
In particular, V is randomly initialized with a Gaussian distribution of mean 0 and standard deviation 1, which follows the standard settings. For the constraints in Eq. 11, for each dataset the centers are formed of 300 points obtained by K-means clustering. We also give experimental analysis on whether the number of centers affects the retrieval performance in Fig. 4 (a). It is worth to note that centers generated by K-means reflect the distribution and structure of data points, which can approximate the original ordinal graph without performance reduction. For the SVD dimension, we set 16 to the parameter d_{svd}, which can get the competitive performance by reducing data noise. The number of centers generated by K-means Clustering is set to 300, and the corresponding analysis will be given in below section.

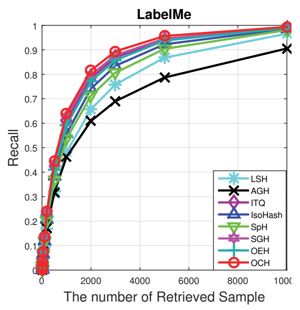
Quantitative Results

As shown in Tab. 1 and Fig. 3 with hash code varied from 32 to 128, the proposed OCH consistently achieves superior

[3]The source codes of all the above methods are provided by authors kindly.


(a) mAP curves on GIST1M.


(b) Rec@K curves on GIST1M.


(c) Rec@K on LabelMe.

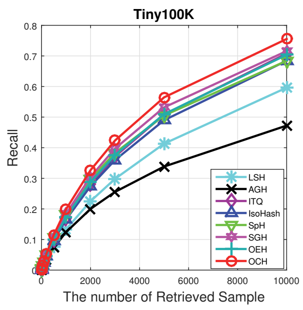

(d) Rec@K on Tiny100K.

Figure 3: The mAP and Recall Curves of all the algorithms on three benchmarks. ((b)-(d) are all evaluated on 64 bit.)

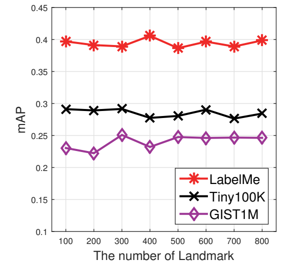
Table 2: The Training Time (s) comparing with different algorithms on both datasets.

	LabelMe		Tiny100K		GIST1M	
Methods	32	64	32	64	32	64
LSH	0.01	0.01	0.04	0.04	0.01	0.01
ITQ	0.47	1.02	0.46	0.78	0.64	0.97
SGH	1.61	3.25	1.62	3.21	1.81	4.43
OEH	49.00	83.79	48.38	80.90	62.2 4	115.53
OCH	29.07	35.01	28.89	35.93	29.11	36.69

performance over all baselines among all datasets, especially when the hash bit is short.

Most previous hashing works always quantize the hash code by minimizing the loss between Euclidean distance and Hamming distance. But in OCH, we change to minimize the inconsistency between original loss before and after learning binary codes. Compared to the previous works, it is quantitatively demonstrated that preserving such ordinal cues in hashing is a more fundamental goal for nearest neighbor search. As for the comparison between our work and the most recent works in ranking preserve hashing OEH (Liu et al. 2016), in comparison, both OEH and OCH adopt a two-step projection to find the optimal binary quantization space. For OEH, the PCA dimension reduction is used as the first step, which cannot reduce the scale of training and still needs about nL^2 triplet order tuples to train the overall model. On the contrary for the proposed OCH, we use the ordinal constraint projection in the first step, which not only reduces the original feature dimension, but also reduces the scale of training. Tab. 2 further shows the comparison of training time between the above methods, in which OCH costs about half training time of OEH but achieves better performance on all the three datasets. Moreover, OEH needs to construct the ordinal relations during each iteration, which is time consuming. In contrast, our OCH constructs the ordinal relations by TOG only once before iteration, which is very convenient and efficient in iterative training.

At last, we discuss the influence of centers and the number of relations used in each iteration. Note that the centers are generated by K-means, and the final mAP result in Fig. 4 (a) is shown with the number of centers increasing from 100


(a) mAP vs. #centers.

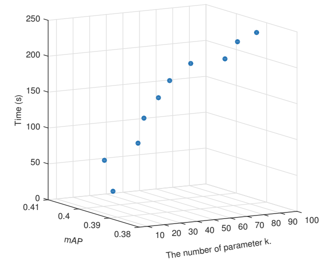

(b) mAP vs. the number of subset.

Figure 4: The parameter analysis when hash bit is 64.

to 800 among three datasets. We find that the performances rarely change with the number increasing of centers. As a result, only a small number of centers is used to approximate the overall ordinal relations, which is set to 300 for all the datasets. As in Fig. 4 (b), we show the relation between mAP and the training time by increasing the number $k(L-k)$ of randomly selected ordinal relations during each iteration. The result shows that the training time is linear to the number of ordinal relations, but the mAP does not change too much. Therefore, we can use a small set of ordinal relations in each iteration, while maintaining the overall search precision.

Conclusion

In this paper, we proposed a novel unsupervised hashing approach, dubbed Ordinal Constraint Hashing (OCH), for large-scale similarity retrieval. Unlike most previous unsupervised hashing, the proposed approach exploits the ordinal information between data points, and embeds such relations into a Hamming space. Firstly, a tensor ordinal graph was proposed to approximate the ordinal relations efficiently. Then, OCH adopted an ordinal constraint projection scheme to significantly reduce the scale of ordinal graph, which preserves the overall ordinal relation through a small centers set (such as K-means centers). In optimization, a novel iterative stochastic gradient descent algorithm on Stiefel manifold was developed. Extensive experiments on three benchmark datasets demonstrated that the proposed OCH approach achieves the best performance in contrast with representative and state-of-the-art hashing methods. In our future work, we will further extend the proposed method with deep learning, as well as investigating the possibility of large-scale binarized optimization in binary code learning.

Acknowledgement

This work is supported by the National Key RD Program (No. 2016YFB1001503), the Special Fund for Earthquake Research in the Public Interest No.201508025, the Nature Science Foundation of China (No. 61402388, No. 61422210, No. 61373076, and No. 61572410), the CCF-Tencent Open Research Fund, the Open Projects Program of National Laboratory of Pattern Recognition, and the Xiamen Science and Technology Project (No. 3502Z20153003).

References

Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2007. *Optimization algorithms on matrix manifolds*. Princeton University Press.

Babenko, A.; Arandjelović, R.; and Lempitsky, V. 2016. Pairwise quantization. *arXiv*.

Cunningham, J. P., and Ghahramani, Z. 2015. Linear dimensionality reduction: Survey, insights, and generalizations. *JLMR*.

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of ASCG*.

Dean, T. L.; Ruzon, M. A.; Segal, M.; Shlens, J.; Vijayanarasimhan, S.; and Yagnik, J. 2013. Fast, accurate detection of 100, 000 object classes on a single machine. In *Proceedings of CVPR*.

Ge, T.; He, K.; Ke, Q.; and Sun, J. 2014. Optimized product quantization. *IEEE TPAMI*.

Gong, Y.; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE TPAMI*.

He, K.; Wen, F.; and Sun, J. 2013. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of CVPR*.

Heo, J.-P.; Lee, Y.; He, J.; Chang, S.-F.; and Yoon, S.-E. 2015. Spherical hashing: binary code embedding with hyperspheres. *IEEE TPAMI*.

Jegou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *IEEE TPAMI*.

Jiang, Q.-Y., and Li, W.-J. 2015. Scalable graph hashing with feature transformation. In *Proceedings of IJCAI*.

Kong, W., and Li, W.-J. 2012. Isotropic hashing. In *Proceedings of NIPS*.

Kulis, B., and Darrell, T. 2009. Learning to hash with binary reconstructive embeddings. In *Proceedings of NIPS*.

Li, P.; Shrivastava, A.; Moore, J. L.; and Konig, A. C. 2011. Hashing algorithms for large-scale learning. In *Proceedings of NIPS*.

Li, X.; Lin, G.; Shen, C.; Van Den Hengel, A.; and Dick, A. R. 2013. Learning hash functions using column generation. In *Proceedings of ICML*.

Lin, C.; Chen, W.; Qiu, C.; Wu, Y.; Krishnan, S.; and Zou, Q. 2014. Libd3c: Ensemble classifiers with a clustering and dynamic selection strategy. *Neurocomputing*.

Lin, G.; Shen, C.; and Wu, J. 2014. Optimizing ranking measures for compact binary code learning. In *Proceedings of ECCV*.

Liu, W.; Wang, J.; Kumar, S.; and Chang, S.-F. 2011. Hashing with graphs. In *Proceedings of ICML*.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.-G.; and Chang, S.-F. 2012a. Supervised hashing with kernels. In *Proceedings of CVPR*.

Liu, W.; Wang, J.; Mu, Y.; Kumar, S.; and Chang, S.-F. 2012b. Compact hyperplane hashing with bilinear functions. *Proceedings of ICML*.

Liu, W.; Mu, C.; Kumar, S.; and Chang, S.-F. 2014. Discrete graph hashing. In *Proceedings of NIPS*.

Liu, H.; Ji, R.; Wu, Y.; and Liu, W. 2016. Towards optimal binary code learning via ordinal embedding. In *Proceedings of AAAI*.

Norouzi, M., and Fleet, D. J. 2011. Minimal loss hashing for compact binary codes. In *Proceedings of ICML*.

Norouzi, M.; Fleet, D. J.; and Salakhutdinov, R. 2012. Hamming distance metric learning. In *Proceedings of NIPS*.

Oliva, A., and Torralba, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*.

Park, Y.; Cafarella, M.; and Mozafari, B. 2015. Neighbor-sensitive hashing. *Proceedings of the VLDB Endowment*.

Shen, F.; Shen, C.; Liu, W.; and Shen, H. T. 2015. Supervised discrete hashing. *Proceeding of CVPR*.

Song, D.; Liu, W.; Ji, R.; Meyer, D. A.; and Smith, J. R. 2015. Top rank supervised binary coding for visual search. In *Proceedings of ICCV*.

Torralba, A.; Fergus, R.; and Weiss, Y. 2008. Small codes and large image databases for recognition. In *Proceedings of CVPR*.

Wang, J.; Liu, W.; Sun, A. X.; and Jiang, Y.-G. 2013a. Learning hash codes with listwise supervision. In *Proceedings of ICCV*.

Wang, J.; Liu, W.; Sun, A. X.; and Jiang, Y.-G. 2013b. Learning hash codes with listwise supervision. In *Proceedings of ICCV*.

Wang, J.; Liu, W.; Kumar, S.; and Chang, S.-F. 2016. Learning to hash for indexing big data - a survey. *Proceedings of the IEEE*.

Wang, J.; Kumar, S.; and Chang, S.-F. 2012. Semi-supervised hashing for large-scale search. *IEEE TPAMI*.

Weinberger, K. Q.; Dasgupta, A.; Langford, J.; Smola, A. J.; and Attenberg, J. 2009. Feature hashing for large scale multitask learning. In *Proceedings of ICML*.

Weiss, Y.; Torralba, A.; and Fergus, R. 2008. Spectral hashing. In *Proceedings of NIPS*.

Wen, Z., and Yin, W. 2013. A feasible method for optimization with orthogonality constraints. *Math. Program.*