Consensus hashing

Cong Leng¹ · Jian Cheng¹

Received: 4 August 2014 / Accepted: 6 April 2015 / Published online: 5 May 2015
© The Author(s) 2015

Abstract Hashing techniques have been widely used in many machine learning applications because of their efficiency in both computation and storage. Although a variety of hashing methods have been proposed, most of them make some implicit assumptions about the statistical or geometrical structure of data. In fact, few hashing algorithms can adequately handle all kinds of data with different structures. When considering hybrid structure datasets, different hashing algorithms might produce different and possibly inconsistent binary codes. Inspired by the successes of classifier combination and clustering ensembles, in this paper, we present a novel combination strategy for multiple hashing results, named consensus hashing. By defining the measure of consensus of two hashing results, we put forward a simple yet effective model to learn consensus hash functions which generate binary codes consistent with the existing ones. Extensive experiments on several large scale benchmarks demonstrate the overall superiority of the proposed method compared with state-of-the-art hashing algorithms.

1 Introduction

Nearest neighbor (NN) search is a fundamental building block in many machine learning and computer vision applications, such as image recognition, image matching, collaborative filtering, etc. It becomes infeasible for exhaustive linear search when the data explosively increases in size and dimensionality. To overcome the high time complexity in NN search, many approximate nearest neighbor (ANN) search methods have been proposed ([Arya et al.](#)

Editors: João Gama, Indrè-Žliobaitė, Alípio M. Jorge, and Concha Bielza.

✉ Jian Cheng
jcheng@nlpr.ia.ac.cn

Cong Leng
cong.leng@nlpr.ia.ac.cn

¹ National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China

1998; Friedman et al. 1977; Silpa-Anan and Hartley 2008; Indyk and Motwani 1998). Among these techniques, hashing based ANN search has attracted much attention recently (Indyk and Motwani 1998; Datar et al. 2004). Hashing algorithms encode nearby points in the original space into similar codes in a discrete binary space, i.e., Hamming space. This enables large efficiency gains in computation speed for similarity search. For example, calculating the Hamming distance between 64-bit codes costs only one CPU cycle with the XOR operation.

Numerous hashing methods have been proposed over the last decade, which can be roughly categorized into *data independent* and *data dependent* schemes. Data independent hashing methods, such as the well known locality sensitive hashing (LSH) (Indyk and Motwani 1998; Charikar 2002) and its variants (Kulis and Grauman 2009; Datar et al. 2004; Jain et al. 2008), construct hash functions based on random projections without considering the input data. Rather than random projection, data dependent methods try to learn data-aware hash functions from a training set. Representative approaches include spectral hashing (SH) (Weiss et al. 2008), anchor graph hashing (AGH) (Liu et al. 2011), binary reconstructive embedding (BRE) (Kulis and Darrell 2009), iterative quantization (ITQ) (Gong et al. 2013), spherical hashing (SPH) (Heo et al. 2012), K-means Hashing (KMH) (He et al. 2013), etc. Supervised information is also incorporated into newly proposed hashing methods (Wang et al. 2012; Norouzi and Fleet 2011; Liu et al. 2012; Lin et al. 2013), and it has been demonstrated that side information is conducive to preserving the semantic affinity.

Although a variety of hashing methods have been proposed, most of them make some assumptions about the statistical or geometrical structure of data. For instance, SH assumes that the data is uniformly distributed (Weiss et al. 2008), while AGH assumes that the data lies on a manifold (Liu et al. 2011). In fact, few hashing algorithms can adequately handle all kinds of data with different structures. When considering real-world datasets of hybrid structure, different hashing methods produce different and possibly inconsistent hashing results. Two data points might be embedded to close codes by one method but far by another. Actually, even multiple replications of the same algorithm, such as LSH, result in different hashing codes due to the algorithm's inner randomness. On the other hand, in many visual applications, the data can be represented by different kinds of features, and it is clear that different hashing codes will be obtained when one algorithm is applied on different features. In all these cases, the diverse outcomes usually contain complementary information which could be useful to improve the performance of hashing. Inspired by the success of classifier combination (Breiman 1996; Schapire 1990) and clustering ensembles (Fred and Jain 2005) in practice, we believe the idea of combining these complementary information for obtaining better binary codes is interesting and worth investigating.

In this paper, we first study the problem of combining multiple hashing results and put forward a novel strategy, named *Consensus Hashing* (CH). Generally the problem we focus on here is finding a combined hashing result based on multiple existing ones so as to provide better search performance. There exist two major challenges for this problem. First, how to measure the consensus of two hashing results? The measurement of consensus in hashing is more difficult than that in classifier combination (Breiman 1996; Schapire 1990) because two completely consistent hashing results may seem different in appearance. Second, how to learn consensus hash functions? Finding consensus codes for the training data is not sufficient because the algorithm should also be able to generate hashing codes for the unseen queries. It is necessary to learn new consensus hash functions to solve the problem of out-of-sample extension. To overcome these challenges, we first propose a simple approach to measure the consensus of two hashing results based on connectivity matrix. With this measurement, an effective model is established to learn consensus hash functions which generate binary code consistent with the existing ones. We find that the optimization of the model can be relaxed

into two subproblems: a multidimensional scaling (MDS) (Kruskal and Wish 1978) problem and a basic linear regression problem, and closed-form solutions exist for both subproblems. By taking advantage of the Nyström technique (Williams and Seeger 2001), our strategy has linear training time with respect to the size of the training set. One interesting characteristic of our method is that it is inherently qualified for multi-view hashing by combining the diverse hashing codes generated by different features. The effectiveness and efficiency of the proposed CH are validated by extensive experiments, both single-view and multi-view, carried out on several benchmarks.

2 Background

The idea of combining diverse information of multiple outcomes lies in the context of ensemble learning. As a kind of state-of-the-art learning approach, the core spirit of ensemble learning is that one can train multiple learners and then combine them for use. Notable works include Bagging (Breiman 1996), Random Forest (Breiman 2001) and Boosting (Schapire 1990). It is well known that an ensemble is usually more accurate than a single learner, and ensemble methods have already achieved great success in many real-world tasks (Zhou 2012). Abundant theoretical results have been established for these three ensemble learning methods. For example, Breiman proved that Bagging and Random Forest converge to a steady error rate as the ensemble size grows (Breiman 1996; Breiman 2001). Freund and Schapire proved that the error of the final combined learner in AdaBoost is upper bounded by the errors of the base learners (Freund and Schapire 1996).

Consensus learning is another learning strategy of ensemble learning. This learning scheme is originally designed for *unsupervised* tasks such as clustering, i.e., consensus clustering (Monti et al. 2003; Li et al. 2007; Fred and Jain 2005). Consensus clustering is a kind of ensemble whose base learners are clusters generated by clustering methods. Ensemble of clustering is considerably more difficult than the combination of classifiers, because the cluster labels produced by different clustering methods may not correspond. During the past decades, a great number of consensus clustering approaches have been proposed. Representative works include similarity based methods (Fred and Jain 2005), graph based methods (Fern and Brodley 2004) and non-negative matrix factorization (NMF) based methods (Li et al. 2007). Similarity based methods express the base clustering results as similarity matrices and generate the final clustering result based on the consensus similarity matrix (Fred and Jain 2005). The basic idea of graph based methods is to construct an undirected graph to model the base clustering results, and then derive the ensemble clustering via graph partitioning (Fern and Brodley 2004). Li et al. (2007) found that consensus clustering can be formulated within the framework of NMF and proposed a simple iterative algorithm. Recently, weighted consensus clustering (Li and Ding 2008) has been considered in which a weight for each base clustering is introduced to model its importance. Although these consensus clustering methods have shown the ability to improve clustering quality and robustness in practice, unlike other well-established ensemble learning strategies such as Bagging and Boosting, finding theoretical guarantees for the consensus learning strategy is still an open question.

Our work is largely inspired by the works in ensemble learning especially consensus clustering. Different from consensus clustering, the base learners in our work are the diverse binary codes generated by hashing algorithms. Our work aims to look for a combined hashing result based on the existing ones in order to achieve better performance. More specifically, we establish a complete learning system which is equipped with the ability to measure the

consensus of binary codes (Sect. 3), generate hashing codes for the unseen samples (Sect. 4.1), and overcome the large scale learning issue (Sect. 4.2).

3 Measure of consensus in hashing

As a kind of ensemble learning approach, it is crucial for consensus learning to measure the consensus of different base learners. For classifier combination, it is not a problem because the outcome of each weak classifier is explicit, i.e., positive or negative. Nevertheless, it is not suitable for hashing combination. As we will see later, two consistent hashing results may be entirely different in appearance. It is necessary to seek a new metric to measure the consensus in hashing.

First of all, some notations are described here. Let $X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{d \times n}$ denote a set of n data points. Suppose there exists a set of L component hashing results $\mathcal{H} = \{H^{(1)}, H^{(2)}, \dots, H^{(L)}\}$, each $H^{(l)} = [\text{code}^l(x_1), \dots, \text{code}^l(x_n)]^T \in \{-1, 1\}^{n \times r}$ denotes a code matrix of X , corresponding to the l th component. Each row of $H^{(l)}$ represents the r -bit hashing code of a sample.

In order to give an illustration about the difficulty in measuring the consensus of hashing codes, a toy example is presented as follows: suppose we have only three samples x_1, x_2, x_3 and two hashing results $H^{(p)}, H^{(q)}$ are derived:

$$H^{(p)} = \begin{bmatrix} +1 & +1 & +1 & -1 \\ +1 & -1 & +1 & -1 \\ -1 & -1 & +1 & -1 \end{bmatrix}, \quad H^{(q)} = \begin{bmatrix} -1 & -1 & -1 & +1 \\ -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & +1 \end{bmatrix}$$

The i th row is the binary code of sample x_i . From the code matrix view, these two hashing codes are entirely different. However, by looking closely, it can be found that $H^{(q)} = -H^{(p)}$. That is to say, the two hashing results are actually identical in terms of the Hamming distance between every two samples. In practice, even the code length r of various components could be different, which brings more difficulties for measuring the consensus of different hashing results. In this section, we propose a very simple yet effective measure based on the connectivity matrix.

Connectivity matrix

For each component hashing codes $H^{(l)}$, the $n \times n$ Gram matrix $G^{(l)}$ can be defined as

$$G_{ij}^{(l)} = (\text{code}^l(x_i), \text{code}^l(x_j)).$$

where $(\cdot, \cdot)$ represents inner product. It is easy to find that $(\text{code}^l(x_i), \text{code}^l(x_j)) \in [-r, r]$, which can be normalized into the range $[-1, 1]$ by

$$M_{ij}^{(l)} = G_{ij}^{(l)} / r.$$

$M_{ij}^{(l)} \in [-1, 1]$ can be viewed as a normalized similarity between x_i and x_j in Hamming space (Liu et al. 2012), and

$$M^{(l)} = \frac{1}{r} H^{(l)} H^{(l)T}. \quad (1)$$

In this way, the code length r is allowed to be different in various component hashing results. M is named as *connectivity matrix*.

Measure of consensus

In this paper, we measure the consensus of two hashing results by considering whether two samples are close (or far) in Hamming space in both results. For example, if two samples are close in both two results, then the two results are consistent on these two samples. Otherwise, if two samples are close in one result but far in the other, then the two hashing results are not consistent. As mentioned above, the elements in the connectivity matrix can be used to measure the closeness of two samples in the Hamming space. As a consequence, the connectivity matrix can be used to define the measure for consensus. Concretely, the distance between two hashing codes $H^{(p)}$ and $H^{(q)}$ is defined as:

$$\begin{aligned}\mathcal{D}(H^{(p)}, H^{(q)}) &= d(M^{(p)}, M^{(q)}) \\ &= \|M^{(p)} - M^{(q)}\|_F^2,\end{aligned}\quad (2)$$

where $\|\cdot\|_F$ represents the *Frobenius norm*. The disagreement between two hashing results can be measured by $d(M^{(p)}, M^{(q)})$. When $d(M^{(p)}, M^{(q)}) = 0$, we consider that two hashing results are completely consistent.

For the previous toy example, with Eq. (1), we have the connectivity matrices of $H^{(p)}$ and $H^{(q)}$ as:

$$M^{(p)} = \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix}, \quad M^{(q)} = \begin{bmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix}$$

which are identical, so the two hashing results are completely consistent.

4 Consensus hash functions learning

In this section, we give the details of the proposed Consensus hashing framework. A simple model associated with an efficient two-step optimization method is proposed for consensus hash functions learning.

4.1 Objective function

With the definition of consensus measurement in hashing, we aim to find consensus binary code H which is consistent with the existing ones. The average distance between H and all existing codes is:

$$\begin{aligned}\frac{1}{L} \sum_{l=1}^L \mathcal{D}(H, H^{(l)}) &= \frac{1}{L} \sum_{l=1}^L d(M, M^{(l)}) \\ &= \frac{1}{L} \sum_{l=1}^L \left\| \frac{HH^T}{r} - M^{(l)} \right\|_F^2,\end{aligned}\quad (3)$$

where $M^{(1)}, M^{(2)}, \dots, M^{(L)}$ are the connectivity matrices of component hashing codes $H^{(1)}, H^{(2)}, \dots, H^{(L)}$, respectively. $M = \frac{HH^T}{r}$ is defined as the connectivity matrix of the consensus hashing H . Naturally, the objective function becomes:

$$\begin{aligned}
\min_H \quad & \frac{1}{L} \sum_{l=1}^L \left\| \frac{HH^T}{r} - M^{(l)} \right\|_F^2 \\
\text{s.t.} \quad & H \in \{-1, 1\}^{n \times r}
\end{aligned} \tag{4}$$

For any matrices A and B of the same dimensions, we have $\|A - B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 - 2\text{tr}(A^T B)$. With simple mathematical derivation, it is easy to find that:

$$\begin{aligned}
\frac{1}{L} \sum_{l=1}^L \left\| \frac{HH^T}{r} - M^{(l)} \right\|_F^2 &= \frac{1}{L} \sum_{l=1}^L \|M - M^{(l)}\|_F^2 \\
&= \frac{1}{L} \sum_{l=1}^L \left(\|M\|_F^2 - 2\text{tr}(M^T M^{(l)}) + \|M^{(l)}\|_F^2 \right) \\
&= \|M\|_F^2 - 2 \frac{1}{L} \sum_{l=1}^L \text{tr}(M^T M^{(l)}) + \frac{1}{L} \sum_{l=1}^L \|M^{(l)}\|_F^2 \\
&= \left(\|M\|_F^2 - 2\text{tr}\left(M^T \frac{1}{L} \sum_{l=1}^L M^{(l)}\right) + \left\| \frac{1}{L} \sum_{l=1}^L M^{(l)} \right\|_F^2 \right) - \left\| \frac{1}{L} \sum_{l=1}^L M^{(l)} \right\|_F^2 + \frac{1}{L} \sum_{l=1}^L \|M^{(l)}\|_F^2 \\
&= \left\| M - \frac{1}{L} \sum_{l=1}^L M^{(l)} \right\|_F^2 - \left\| \frac{1}{L} \sum_{l=1}^L M^{(l)} \right\|_F^2 + \frac{1}{L} \sum_{l=1}^L \|M^{(l)}\|_F^2 \\
&= \left\| \frac{HH^T}{r} - \frac{1}{L} \sum_{l=1}^L M^{(l)} \right\|_F^2 + \frac{1}{L} \sum_{l=1}^L \|M^{(l)}\|_F^2 - \left\| \frac{1}{L} \sum_{l=1}^L M^{(l)} \right\|_F^2.
\end{aligned} \tag{5}$$

Since both $\frac{1}{L} \sum_{l=1}^L \|M^{(l)}\|_F^2$ and $\left\| \frac{1}{L} \sum_{l=1}^L M^{(l)} \right\|_F^2$ are constants, CH takes a form of the following optimization problem:

$$\begin{aligned}
\min_H \quad & \left\| HH^T - \frac{r}{L} \sum_{l=1}^L M^{(l)} \right\|_F^2 \\
\text{s.t.} \quad & H \in \{-1, 1\}^{n \times r}
\end{aligned} \tag{6}$$

So far we have only considered how to combine existing multiple hashing results for the training data. However, this is not sufficient because we should also be able to generate hashing codes for the unseen queries. In order to deal with the out-of-sample extension problem, hash functions should be explicitly incorporated into the objective. For the simplicity of implementation for large scale datasets, following Wang et al. (2012), we use linear projection coupled with mean threshold as a hash function. Specifically, the k th hash function is defined as:

$$h_k(\mathbf{x}) = \text{sgn}(w_k^T \mathbf{x} + b_k), \tag{7}$$

where $\mathbf{x}$ is a sample, w_k is a projection vector and b_k is the negative mean of projected data. Without loss of generality, the data X is preprocessed to have zero mean, then $b_k = -\frac{1}{n} \sum_{i=1}^n w_k^T x_i = 0$.

Let $W = [w_1, w_2, \dots, w_r] \in \mathbb{R}^{d \times r}$ be a matrix including a sequence of hashing projection vectors, then the code matrix H can be written as:

$$H = \text{sgn}(X^T W). \tag{8}$$

After substituting H in Eq. (6) with Eq. (8), we arrive at the final objective:

$$\min_{W \in \mathbb{R}^{d \times r}} \left\| \text{sgn}(X^T W) \text{sgn}(X^T W)^T - rU \right\|_F^2, \quad (9)$$

where $\text{sgn}(\cdot)$ is the sign function and

$$U = \frac{1}{L} \sum_{l=1}^L M^{(l)}. \quad (10)$$

It is worth noting that, for a new query data, we only need to apply the learned consensus hash functions W on it to get its hashing code. In other words, the combination of component hashing results is only conducted in the training process.

4.2 Optimization

The objective function given in Eq. (9) is difficult to be optimized because of the discontinuous sign function. In this work, we decompose this problem into two sub-problems and propose a two-step optimization method to solve them. At first, Eq. (9) can be equivalently rewritten as:

$$\begin{aligned} \min_{W \in \mathbb{R}^{d \times r}, Y} \quad & \left\| Y Y^T - rU \right\|_F^2 \\ \text{s.t.} \quad & Y = \text{sgn}(X^T W) \end{aligned} \quad (11)$$

which can be further relaxed into the following two sub-problems:

$$\min_{Y \in \mathbb{R}^{n \times r}} \left\| Y Y^T - rU \right\|_F^2, \quad (12a)$$

$$\min_{W \in \mathbb{R}^{d \times r}} \left\| Y - \text{sgn}(X^T W) \right\|_F^2. \quad (12b)$$

The relaxed variable Y in (12a) is of real space, then (12a) becomes a MDS problem (Kruskal and Wish 1978). Besides, it will be shown that (12b) can be transformed into a basic linear regression problem and a classical Orthogonal Procrustes (OP) problem (Schönmeyer 1966). Therefore, the complicated objective in Eq. (9) can be optimized by solving two relatively simpler problems (12a) and (12b).

4.2.1 First step

Note that the U in Eq. (12a) is a positive semi-definite matrix because each component $M^{(l)}$ is positive semidefinite. Besides, since the relaxed Y is of real space, the first subproblem (12a) becomes a classical MDS problem (Kruskal and Wish 1978) and its global optimum solution can be obtained by eigen-decomposition of the $n \times n$ matrix rU . Concretely, we solve the eigenvalue system of rU , resulting in the top r ($\ll n$) largest eigenvalues $\lambda_1 > \lambda_2 > \dots > \lambda_r > 0$ and their corresponding eigenvectors $v_1, v_2, \dots, v_r$, then the global optimum solution of (12a) is:

$$Y^* = \left[\sqrt{\lambda_1} v_1, \sqrt{\lambda_2} v_2, \dots, \sqrt{\lambda_r} v_r \right]. \quad (13)$$

However, the time complexity of eigen-decomposition for a $n \times n$ matrix is $O(n^3)$, which is infeasible if not impossible for large scale dataset. In order to address this problem, we

propose to use the Nyström technique (Williams and Seeger 2001) to construct a low-rank approximation matrix $\widehat{U}$ for U . In detail, if a subset of the training data of size $m \ll n$ is used to create the connectivity matrices, we can get a $m \times m$ submatrix of U with Eq. (10), denoted as $U_{m,m}$. In addition, the corresponding m columns of U can form a $n \times m$ submatrix of U , denoted as $U_{n,m}$, which is the connectivity matrix of the whole n training data with the m training data in the subset. According to Williams and Seeger (2001), a low-rank approximation matrix for U is

$$\widehat{U} = U_{n,m} U_{m,m}^{-1} U_{n,m}^T. \quad (14)$$

The corresponding approximate eigenvalues and eigenvectors are (Williams and Seeger 2001):

$$\widehat{\lambda}_i = \frac{n}{m} \lambda_i^{(m)}, \quad i = 1, 2, \dots, r \quad (15a)$$

$$\widehat{v}_i = \sqrt{\frac{m}{n}} \frac{1}{\lambda_i^{(m)}} U_{n,m} v_i^{(m)}, \quad i = 1, 2, \dots, r. \quad (15b)$$

where $\lambda_i^{(m)}$ and $v_i^{(m)}$ are the i^{th} eigenvalue and eigenvector of $m \times m$ submatrix $U_{m,m}$. Plugging Eqs. (15a, 15b) into Eq. (13), the approximate optimum solution can be obtained with:

$$Y^* = \left[\frac{1}{\sqrt{\lambda_1^{(m)}}} U_{n,m} v_1^{(m)}, \frac{1}{\sqrt{\lambda_2^{(m)}}} U_{n,m} v_2^{(m)}, \dots, \frac{1}{\sqrt{\lambda_r^{(m)}}} U_{n,m} v_r^{(m)} \right]. \quad (16)$$

This way, the time complexity of eigen-decomposition is reduced from $O(n^3)$ to $O(m^3)$ with $m \ll n$. Nevertheless, it will lead to the unbalance problem if this optimum Y^* is directly used because most of the information is contained in the most significant eigenvectors while the remainders are often noisy (Gong et al. 2013; Kong and Li 2012). Obviously, if Y^* is an optimum solution of problem (12a), then so is $Y^* R$ with R as an arbitrary orthogonal matrix ($RR^T = R^T R = I$) because

$$Y^* R (Y^* R)^T = Y^* R R^T Y^{*T} = Y^* Y^{*T},$$

$Y^* R$ can be viewed as applying a rotation to Y^* . As indicated in Gong et al. (2013), Kong and Li (2012), this rotation is very important for eigen-decomposition based hashing methods to relieve the unbalance problem. In this work, we incorporate such a rotation matrix R in the second step.

4.2.2 Second step

With an orthogonal matrix R incorporated, the subproblem (12b) can be rewritten as:

$$\begin{aligned} \min_{W \in \mathbb{R}^{d \times r}, R \in \mathbb{R}^{r \times r}} & \quad \|Y^* R - \text{sgn}(X^T W)\|_F^2 \\ \text{s.t.} & \quad RR^T = I \end{aligned} \quad (17)$$

This objective is not convex with respect to W or R . The optimization process can be conducted with two alternating steps:

(1) Update W with fixed R :

As R is fixed, $Y^* R$ is fixed and the constraint is gone. The objective is minimized when

$$\text{sgn}(X^T W) = \text{sgn}(Y^* R).$$

Algorithm 1 Consensus Hashing (CH)

Input: Existing binary codes $H^{(l)}, l = 1, 2, \dots, L$, a training set $X \in \mathbb{R}^{d \times n}$, code length r .

Output: Hash functions $W = [w_1, w_2, \dots, w_r]$.

First step

- 1: Randomly sample m indices to generate submatrices $U_{n,m}$ and $U_{m,m}$.
- 2: Do eigendecomposition to $U_{m,m}$, obtaining the top r largest eigenvalue $\lambda_i^{(m)}$ and the corresponding eigenvectors $v_i^{(m)}$.
- 3: Compute Y^* with Eq. (16).

Second step

- 4: Randomly initialize R .
- 5: **while** Eq. (17) is not converged **do**
- 6: Update W with Eq. (18);
- 7: Update R with Eq. (19);
- 8: **end while**
- 9: Output hash functions W .

An obvious optimum solution W exists if it satisfies

$$X^T W = Y^* R.$$

This becomes a basic linear regression problem and the optimum solution is

$$W = (XX^T + \sigma I)^{-1} XY^* R, \quad (18)$$

where σ is a very small positive constant.

(2) Update R with fixed W :

When W is fixed, the objective is similar to that in ITQ (Gong et al. 2013) in form but with different meaning. For a fixed $\text{sgn}(X^T W)$, the objective (17) becomes a classical Orthogonal Procrustes problem (Schönmeyer 1966) which can be solved with SVD decomposition. More specifically, compute the SVD of $\text{sgn}(X^T W)^T Y^*$ as $S\Sigma\hat{S}^T$ and then the optimum R is

$$R = \hat{S}S^T. \quad (19)$$

In the experiments, we find that 30–50 iterations is usually enough to ensure the algorithm converge. The whole flowchart of the proposed CH is shown in Algorithm 1.

4.3 Complexity analysis

The complexity of eigendecomposition for a $m \times m$ matrix $U_{m,m}$ is $O(m^3)$. To obtain Y^* in Eq. (16), we need to multiply $n \times m$ matrix $U_{n,m}$ with the r eigenvectors, which costs $O(rmn)$ complexity. Therefore, the time complexity of first step is $O(m^3 + rmn)$. In Eq. (18), in order to obtain $(XX^T)^{-1}$, the time complexity is $O(dn + d^3)$, and the following matrix multiplication costs $O(dn + dr^2 + d^2r)$, then the time complexity for obtaining W is $O(2dn + dr^2 + d^2r + d^3)$. To obtain R in Eq. (19), SVD is applied to a $r \times r$ matrix whose time complexity is $O(r^3)$. Considering the cost to get this matrix $O(r^2n)$, the whole cost for obtaining R is $O(r^3 + r^2n)$. Given that $n \gg r$ and $n \gg d$ in large scale dataset, the whole time complexity of the training process is $O(r^2n + drn)$, which is linear to the size of training set.

5 Experiments

As we have discussed in the introduction, many possible circumstances can lead to different and inconsistent binary codes of the data, including (1) applying different hashing algorithms to the data, (2) the dependency to the initialization or inner randomness of some algorithms, and (3) applying one algorithm to different features of the data. In this section, we conduct extensive experiments to test the effectiveness and efficiency of the proposed CH in all these situations.

5.1 Experiments on three large scale datasets

5.1.1 Datasets

Three large scale datasets are used in the experiments. **Tiny-100K**: it consists of 100,000 tiny images of 32×32 pixels randomly sampled from the original 80 million tiny images.¹ Each image is represented by a 384 dimensional GIST descriptor (Oliva and Torralba 2001). **CIFAR10**:² it consists of 60,000 color images and we extract a 512 dimensional GIST descriptor to represent each image. **GIST-1M**:³ it contains a set of 960 dimensional, one million GIST descriptors extracted from random images.

All these data are mean-centered as required in many methods (Gong et al. 2013; Kong and Li 2012; Kulis and Darrell 2009). For each dataset, we randomly select 1000 data points as queries and use the remaining as gallery database as well as training set. The ground truth neighbors, exact K -nearest neighbors for each query, are computed by linear scan, i.e., comparing each query with all the points in the database with raw feature. Following Wang et al. (2012), the top 2 percentile nearest neighbors in Euclidean space are taken as ground truth. All the reported results are averaged over ten random test/training partitions.

5.1.2 Compared methods and protocol

We compare the performance of our approach, CH, against several state-of-the-art hashing methods, including LSH (Charikar 2002, AGH Liu et al. 2011, ITQ Gong et al. 2013), Isotropic Hashing (IsoH) (Kong and Li 2012) and Kmeans Hashing (KMH) (He et al. 2013). All the results were obtained with the source codes generously provided by the authors and by following their instructions to tune the algorithm parameters.

To perform fair evaluation, we adopt the Hamming Ranking search strategy commonly used in the literature (Gong et al. 2013; Wang et al. 2012; He et al. 2013; Kong and Li 2012). All points in the database are ranked according to their Hamming distance to the query. The Hamming ranking performance is measured with three widely used metrics in information retrieval: mean average precision (MAP), precision-recall curves and precision curves. It is worthy to note that Hamming ranking is an exhaustive linear search method, but usually very fast in practice because of the efficient computation of Hamming distance. Hamming ranking can be further sped up by a recent method (Norouzi et al. 2014). This work of Norouzi et al. (2014) provides a parallel method for fast Hamming ranking in Hamming space, and can be also applied in our work for parallel search. In specific, the code generated by our approach

¹ <http://groups.csail.mit.edu/vision/TinyImages/>.

² <http://www.cs.toronto.edu/~kriz/cifar.html>.

³ <http://corpus-texmex.irisa.fr/>.

can be divided into many pieces of substrings and used for parallel exact k-nearest neighbor search in Hamming space with the approach proposed in [Norouzi et al. \(2014\)](#).

For the proposed consensus hashing, we present two different implementations in the comparisons. The first implementation, denoted as CH^1 , combines the hashing codes generated by 20 independent runs of LSH to learn consensus hash functions. The second implementation, denoted as CH^2 , combines the hashing results of the adopted baselines, i.e., LSH ([Charikar 2002](#)), AGH ([Liu et al. 2011](#)), ITQ ([Gong et al. 2013](#)), IsoH ([Kong and Li 2012](#)) and KMH ([He et al. 2013](#)).

CH^1 combines the binary codes generated by multiple independent runs of LSH because only LSH, owing to its inner randomness, can guarantee enough diversity of the outputs of multiple executions. For other base learner like ITQ, although with better quality, the outputs of different runs are almost the same, which is not suitable for CH^1 . CH^2 aggregates the results generated from various base learners. Ideally, if the diversity of combined learners is sufficient, the higher precision of them will lead to better performance of CH^2 . Nevertheless, in practice, higher precision often means less diversity. This is also why often weak classifiers are used to be base learners in other ensemble learning methods like boosting. In summary, the selection of base combined methods is a trade off between quality and diversity.

5.1.3 Results and analysis

Mean average precision MAP score is one of the most comprehensive criterions to evaluate the retrieval performance in the literature ([Gong et al. 2013](#); [Kong and Li 2012](#); [Heo et al. 2012](#); [He et al. 2013](#)). Figure 1 shows the MAP scores for different methods with different code lengths on three datasets. We observe that CH^1 and CH^2 achieve the highest MAP scores in most cases on all these datasets.

Comparing the data dependent methods with the data independent methods, we find that the data dependent methods like ITQ and IsoH are generally better than the LSH, especially with small code length. As an example, with 16 bits on Tiny-100K, the MAP value of LSH is merely 0.078 while ITQ and IsoH arrive at 0.16 and 0.15 respectively. LSH can therefore be seen as a weak hashing on these datasets. By combining the codes generated by independent runs of LSH, CH^1 obtains a remarkable improvement on LSH, and consistently outperforms other state-of-the-art methods on these datasets. This demonstrates that the performance of weak hashing methods can be largely improved by the proposed consensus strategy, which is consistent with the previous conclusions made in classifier combination ([Schapire 1990](#); [Breiman 1996](#); [Breiman 2001](#)) and clustering combination ([Monti et al. 2003](#); [Li et al. 2007](#); [Fred and Jain 2005](#)).

By comparing CH^2 with the other baselines it combines, i.e. LSH, AGH, ITQ, IsoH and KMH, we find that CH^2 outperforms all of them with a large margin on these datasets. These results imply that the consensus strategy can collect the advantages of other methods and consequently achieve further improvement on any single baseline. Besides, CH^2 consistently outperforms CH^1 . This phenomenon is natural and easy to understand as the information provided by the combined methods is more diverse in CH^2 , e.g., AGH considers the manifold of data and KMH considers the clustering of data. Intuitively, these diverse information is conducive to learning consensus hash functions which are more adequate to capture the structure of data.

Also, some interesting phenomena about the baselines can be observed from the MAP results. AGH works relatively well for small code size and substantially outperforms LSH. However, as the code size increases to 64 bits, the performance of LSH rises rapidly and surpasses AGH. This is due to that most of the information is caught in the top eigenvectors (hash

Fig. 1 Mean average precision (MAP) of different methods on three datasets. **a** MAP with various code lengths on Tiny-100K. **b** MAP with various code lengths on CIFAR10. **c** MAP with various code lengths on GIST-1M. (best viewed in color)

Fig. 2 Comparison results of different hashing methods on the Tiny-100K. **a, b** Precision-recall curves with 64, 128 bits. **c, d** Precision curves with 64, 128 bits. (best viewed in *color*)

functions) while the remainders are usually noisy in AGH. However, in data-independent methods such as LSH, it is theoretically guaranteed that two similar samples will be embedded into close codes with higher probability when more bits are assigned.

Precision-recall curves and precision curves Figures 2–4a, b show the complete precision recall curves on the three datasets, respectively. As a complementary evaluation, the precision curves on three datasets are given in Figs. 2–4c, d. Results with 64 bits and 128 bits are reported. The comparisons with other code lengths are of similar trends. These results demonstrate the overall performance improvement of the proposed CH on other methods. Specially, from these results we can get two observations. First, these detailed results are consistent with the trends discovered in the Fig. 1, namely, our methods perform the best and the running up methods are ITQ and IsoH. Besides, the performance of CH² is better than CH¹ by combining more comprehensive baselines. Second, we find that CH² with 64 bits performs similarly or even better compared with other approaches with 128 bits. In conse-

Fig. 3 Comparison results of different hashing methods on the CIFAR10. **a, b** Precision-recall curves with 64, 128 bits. **c, d** Precision curves with 64, 128 bits. (best viewed in *color*)

quence, our method typically provides about two times more compact binary representation than other methods when meeting the same precision target.

5.1.4 Parametric sensitivity

Although fixed as 20, the number of runs (L) of the weak hashing scheme (i.e., LSH) is a critical parameter in CH¹. We now evaluate the effect of this parameter in our method. Figure 5 compares the MAP scores with 64 bits and 128 bits on Tiny100K and CIFAR10 for different L . In order to observe the effect of parameter L on our algorithm, we vary it from 2 to 30. It can be observed that CH¹ performs better as the number of runs of LSH increases, especially when $L < 20$. However, when L is larger than 20, the performance of our CH¹ tends to be stable. This is in line with what we expected. As L increases, the useful information for CH¹ enlarges, which is conducive to consensus hash function learning. On

Fig. 4 Comparison results of different hashing methods on the GIST-1M. **a, b** Precision-recall curves with 64, 128 bits. **c, d** Precision curves with 64, 128 bits. (best viewed in color)

the other hand, when L becomes larger, the information becomes to be saturated and so that the performance gets to a stable status.

5.1.5 Computational cost

The training time of various methods on Tiny-100K is shown in Table 1. Assuming the combined component hashing codes already exist, the training time of CH¹ and CH² is the same (denoted as CH). We can see that LSH, ITQ, IsoH and our method are very fast, capable to complete the whole training process in several seconds. In terms of time complexity, ITQ, IsoH and our method are all linear to the size of training set. This property is highly beneficial for hashing learning in large scale dataset. In comparison, AGH and KMH need relatively more computational cost. Most of the training time of AGH is spent on the K-means step for obtaining anchor points (62.83 s), although we have selected only a subset of the training

Fig. 5 Mean average precision with different number of independent runs of LSH (L) for CH^1 on **a** Tiny100K and **b** CIFAR10

Table 1 Comparison of training time (seconds) on Tiny-100K

	16-bits	32-bits	48-bits	64-bits	96-bits	128-bits
LSH	0.01	0.01	0.02	0.02	0.03	0.05
AGH	65.81	65.72	65.69	63.52	66.06	65.98
ITQ	2.20	3.71	5.50	7.36	11.51	16.38
IsoH	1.00	0.62	0.79	1.04	0.99	1.18
KMH	428.05	424.20	457.09	487.92	529.97	571.74
CH	2.19	3.60	4.74	6.87	12.37	20.33

data for this step as advised in (Liu et al. 2011). The experiments are carried out on a PC with Intel(R) Core(TM) i5-2400 CPU@3.3GHz and 20GB memory.

5.2 Experiments on multi-view hashing

5.2.1 Multi-view hashing

In most of the real world visual problems, data are collected from diverse domains or obtained from various feature extractors and exhibit heterogeneous properties. Most of the conventional hashing methods, such as all the algorithms we mentioned above, usually adopt a single modality to learn hash functions without exploiting the complementary information contained in different modalities. From this perspective, all these methods can be seen as *single-view hashing*.

Some recently proposed *multi-view hashing* methods try to fuse multiple information sources to get more efficient and effective hashing codes. Song et al. (2011) presented a Multiple Feature Hashing (MFH) to tackle the near-duplicate video retrieval problems. MFH establishes one graph for each view to preserve the local structure information and also globally consider the local structures for all views. Liu et al. (2014) used kernel trick to capture the similarity affinity of different sources (MFKH). In MFKH, by concatenating different features in kernel space, multi-view hashing is formulated as a similarity preserving

problem with linearly combined multiple kernels. Other representative works include Zhang et al. (2011) and Xia et al. (2012).

As pointed out in the introduction, applying the same hashing algorithm on different modalities of the data will result in different hashing codes. These diverse hashing results usually contain complementary information which can be combined with our CH strategy. In this sense, the proposed CH method provides an alternative for multi-view hashing, which is from a completely different perspective of consensus learning compared with the previous methods. In this subsection, we explore the effectiveness of this alternative in multi-view hashing.

5.2.2 Dataset and protocol

We conduct experiments on NUSWIDE⁴ dataset, which consists of 269,648 images with 81 concept tags from Flickr. Five kinds of features are provided and used in our experiments, including 64-D color histogram, 144-D color correlogram, 73-D edge direction histogram, 128-D wavelet texture and 225-D block-wise color moments. To obtain diverse hashing results, we run LSH on each feature ten times. Both MFH and our method need to concatenate all features into a long vector to learn hash functions. For evaluation, we use 1000 images with largest number of tags as queries and the rest serving as database. The source codes are provided by the authors.

For a fair comparison, mean average precision (MAP) and precision of the top K returned examples (Precision@ K) are adopted to evaluate the Hamming ranking. Moreover, we adopt another search method, hash lookup, to evaluate the retrieval performance. In specific, for each query, all potential neighbors within a Hamming radius 2 are retrieved, and then the precision is calculated. We use semantic ground truth to evaluate the performance because the NUSWIDE dataset is associated with semantic tags. The true neighbor is defined based on whether two images share *at least* one common tag.

5.2.3 Results and analysis

Figure. 6 displays the experimental results on NUSWIDE including performances of Hamming ranking and hash lookup. First, Fig. 6a show the MAP scores of different methods with various code lengths. It can be noted that, by combining the hashing results on different features, our CH gives much better performance than the competitors MFH and MFKH. In Fig. 6b, we present the precision of the top 50 returned with different code lengths. Again, significant performance gaps are observed between CH and the two baseline algorithms. Another discovery is that, as the bit number increases, the retrieval precision of MFKH mildly decreases and that of MFH does not increase obviously either. By comparison, our CH achieves remarkably better precision as the code length increases and consistently outperforms other baselines in all cases.

To evaluate the performance of our algorithm in hash lookup, in Fig. 6c we plot the precision of points within Hamming radius 2, i.e. with Hamming distance < 2 to the query, using 16, 32 and 48 bits. Note that we follow Liu et al. (2012) and Wang et al. (2012) to treat failing to find any potential neighbors for a query as zero precision. Due to the increased sparsity of the Hamming space as the code length increases, precision drops rapidly when longer codes are used. It is observed that CH achieves superior accuracy on different bits compared with other methods. It comes to a conclusion that our method with compact codes

⁴ <http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm>.

Fig. 6 Comparison results of different multi-view hashing methods on the NusWide. **a** MAP with various code lengths. **b** Precision @ 50 with various code lengths. **c** Hash lookup precision @ Hamming radius 2 with various code lengths. (best viewed in *color*)

can retrieve more semantically related images than other baselines when using hash lookup. All these results verify that the proposed CH is very effective for multi-view hashing.

6 Conclusion

In this paper, we proposed a novel CH algorithm based on ensemble learning strategy. Firstly, the definition of consensus measurement was proposed. With this definition, we presented a simple model to learn consensus hash function. A two-step optimization method was also proposed for efficient training. Comprehensive analysis certified that the proposed training method has a linear time complexity to the size of training set. Extensive experiments on several large scale benchmarks demonstrated the effectiveness and efficiency of our method.

To the best of our knowledge, this is the first attempt to introduce the idea of consensus learning into hash functions learning. Our work can be viewed as an application of consensus learning in hashing. As another example of consensus learning, consensus clustering is a well studied topic in machine learning community and many interesting approaches have been proposed. Plenty of instructive ideas in consensus clustering are worth studying for hashing. From this point of view, our preliminary work might stimulate other researchers to move their attention to this topic, and finally propose better methods for consensus hashing.

The major limitation of our method is that all the input component hashing codes in our method are treated equally. In real-world cases, however, various component hashing results might be of different importance and some of them might be redundant. Accordingly, controlling the contributions of these results is crucial. In the future work, we intend to extend the consensus hashing model to a weighted scheme, in which different weights will be imposed to different component hashing results and these weights should be learned automatically.

Acknowledgments This work was supported in part by 863 Program (Grant No. 2014AA015100), National Natural Science Foundation of China (Grant No. 61170127, 61332016). The authors would like to thank Prof. Hanqing Lu and Dr. Xi Zhang for their constructive suggestions.

References

- Arya, S., Mount, D., Netanyahu, N., Silverman, R., & Wu, A. (1998). An optimal algorithm for approximate nearest neighbor searching. *Journal of ACM*, 45(6), 891–923.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- Charikar, M. (2002). Similarity estimation techniques from rounding algorithm. In: *ACM symposium on theory of computing* (pp. 380–388).
- Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In: *SCG* (pp. 253–262).
- Fern, X., & Brodley, C. (2004). Solving cluster ensemble problems by bipartite graph partitioning. In: *international conference on machine learning (ICML)*.
- Fred, A., & Jain, A. (2005). Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and machine intelligence* pp. 835–850.
- Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In: *international conference on machine learning (ICML)*.
- Friedman, J., Bentley, J., & Finkel, R. (1977). An algorithm for finding best matches in logarithmic expected time. *TOMS*, 3(3), 209–226.
- Gong, Y., Lazebnik, S., Gordo, A., & Perronnin, F. (2013). Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12), 2916–2929.

- He, K., Wen, F., & Sun, J. (2013). K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In: *IEEE international conference on computer vision and pattern recognition (CVPR)*. IEEE.
- Heo, J.P., Lee, Y., He, J., Chang, S.F., & Yoon, S.E. (2012). Spherical hashing. In: *IEEE international conference on computer vision and pattern recognition (CVPR)*. IEEE.
- Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In: *ACM symposium on theory of computing (STOC)* (pp. 604–613).
- Jain, P., Kulis, B., & Grauman, K. (2008). Fast image search for learned metrics. In: *IEEE international conference on computer vision and pattern recognition (CVPR)*.
- Kong, W., & Li, W. (2012). Isotropic hashing. In: *advances in neural information processing systems (NIPS)*. Kruskal, J. B., & Wish, M. (1978). *Multidimensional scaling* (Vol. 11). New York: Sage.
- Kulis, B., & Darrell, T. (2009). Learning to hash with binary reconstructive embeddings. In: *advances in neural information processing systems (NIPS)*.
- Kulis, B., & Grauman, K. (2009). Kernelized locality-sensitive hashing for scalable image search. In: *IEEE international conference on computer vision (ICCV)*.
- Li, T., & Ding, C. (2008). Weighted consensus clustering. In: *SDM*.
- Li, T., Ding, C., & Jordan, M.I. (2007). Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In: *IEEE international conference on data mining (ICDM)*.
- Lin, G., Shen, C., Suter, D., & Hengel, A.v.d. (2013). A general two-step approach to learning-based hashing. In: *IEEE international conference on computer vision (ICCV)*.
- Liu, W., Wang, J., Ji, R., Jiang, Y., & Chang, S. (2012). Supervised hashing with kernels. In: *IEEE international conference on computer vision and pattern recognition (CVPR)*.
- Liu, W., Wang, J., Kumar, S., & Chang, S. (2011). Hashing with graphs. In: *international conference on machine learning (ICML)*.
- Liu, X., He, J., & Lang, B. (2014). Multiple feature kernel hashing for large-scale visual search. *Pattern Recognition*, 47(2), 748–757.
- Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52(1–2), 91–118.
- Norouzi, M., & Fleet, D.J. (2011). Minimal loss hashing for compact binary codes. In: *international conference on machine learning (ICML)*.
- Norouzi, M., Punjani, A., & Fleet, D. (2014). Fast exact search in hamming space with multi-index hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6), 1107–1119.
- Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3), 145–175.
- Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2), 197–227.
- Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1), 1–10.
- Silpa-Anan, C., & Hartley, R. (2008). Optimised kd-trees for fast image descriptor matching. In: *IEEE conference on computer vision and pattern recognition*.
- Song, J., Yang, Y., Huang, Z., Shen, H.T., & Hong, R. (2011). Multiple feature hashing for real-time large scale near-duplicate video retrieval. In: *proceedings of the 19th ACM international conference on multimedia* (pp. 423–432). ACM.
- Wang, J., Kumar, S., & Chang, S. (2012). Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12), 2393–2406.
- Weiss, Y., Torralba, A., & Fergus, R. (2008). Spectral hashing. In: *advances in neural information processing systems (NIPS)*.
- Williams, C., & Seeger, M. (2001). Using the nyström method to speed up kernel machines. In: *advances in neural information processing systems 13*. Citeseer.
- Xia, H., Wu, P., S.C.Hoi, & Jin, R. (2012). Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In: *proceedings of ACM SIGIR conference (SIGIR)*.
- Zhang, D., Wang, F., & Si, L. (2011). Composite hashing with multiple information sources. In: *proceedings of ACM SIGIR conference (SIGIR)*.
- Zhou, Z. H. (2012). *Ensemble methods: Foundations and algorithms*. Florida: CRC Press.