

Supervised Hashing based on Energy Minimization

Zihao Hu, Xiyi Luo, Hongtao Lu*, and Yong Yu

Department of Computer Science and Engineering, Shanghai Jiao Tong University
{zihao.hu, moberq.luo, htlu}@sjtu.edu.cn, yyu@apex.sjtu.edu.cn

Abstract

Recently, supervised hashing methods have attracted much attention since they can optimize retrieval speed and storage cost while preserving semantic information. Because hashing codes learning is NP-hard, many methods resort to some form of relaxation technique. But the performance of these methods can easily deteriorate due to the relaxation. Luckily, many supervised hashing formulations can be viewed as energy functions, hence solving hashing codes is equivalent to learning marginals in the corresponding conditional random field (CRF). By minimizing the KL divergence between a fully factorized distribution and the Gibbs distribution of this CRF, a set of consistency equations can be obtained, but updating them in parallel may not yield a local optimum since the variational lower bound is not guaranteed to increase. In this paper, we use a linear approximation of the sigmoid function to convert these consistency equations to linear systems, which have a closed-form solution. By applying this novel technique to two classical hashing formulations KSH and SPLH, we obtain two new methods called EM (energy minimizing based)-KSH and EM-SPLH. Experimental results on three datasets show the superiority of our methods.

1. Introduction

Nearest neighbor search (NNS) is a well-known problem arising in numerous fields of application for finding points closest to a given query. Exact nearest neighbor search is intractable in high-dimensional spaces and unnecessary in many cases. Hence, hashing has been merited for it can refine retrieval speed and storage cost considerably [22, 19, 17, 16, 8, 21, 14, 23, 13, 18, 7].

Studies on hashing are roughly in two streams, categorized by whether the learned hash functions rely on the training data. The early exploration of hashing focuses on

using random projections to construct hash functions, thus is data-independent. The most popular data-independent method is Locality Sensitive Hashing (LSH) [2], which is widely used until now.

Data-dependent hashing methods have attracted considerable attention in recent years, since they leverage the training data to achieve better performance. Data-dependent hashing can be divided into two types, i.e., unsupervised and supervised methods. Unsupervised techniques learn underlying linear or non-linear local structures of the training data. Representative methods in this fashion include iterative quantization (ITQ) [3] and locally linear hashing (LLH) [5]. However, in many real-world scenarios, preserving semantic information is more important. Hence, supervised hashing methods are proposed to leverage semantic tags of data points. Representatives of supervised hashing methods include CCA-ITQ [3], kernelized supervised hashing (KSH) [16], two step hashing (TSH) [14], latent factor hashing (LFH) [24], supervised discrete hashing (SDH) [18] and COSDISH [7].

Although many efforts have been made on designing new formulations and optimization procedures for supervised hashing, there are few works that adopt probabilistic inference techniques. LFH [24] might be the first method to use a generative model for supervised hashing, which assumes that the pairwise similarity is generated by the inner product of two corresponding binary codes. By introducing a prior on hashing codes and applying some form of relaxation, the resulting model is easy to optimize and can yield a satisfactory performance. Bayesian supervised hashing (BSH) [4] adopts the mean-field approach to infer latent factors and tune hyper-parameters automatically. However, BSH models each hashing code with a multivariate Gaussian distribution. When learning d -bit codes for n data points, the space complexity is $\mathcal{O}(nd^2)$, which is unbearable for real applications.

In this paper, we propose a novel technique which is also based on probabilistic inference. But instead of modeling the hashing code as a multivariate Gaussian random vector,

*Corresponding author.

we view each binary bit as a discrete random variable that only takes values of 1 and -1 . Therefore, finding the most probable hashing codes is equivalent to solving marginals in the corresponding CRF. We adopt a fully factorized mean-field inference to obtain consistency equations, and approximate these equations by linear systems instead of iterating them, therefore, a closed-form solution can be obtained. We use this technique to improve kernel-based supervised hashing (KSH) [16] and sequential projection learning for hashing (SPLH) [20], and we obtain two new hashing methods called EM (energy minimizing based)-KSH and EM-SPLH. Contributions of this paper are summarized as follows:

- We first view solving different supervised hashing formulations as learning marginals in corresponding CRFs, and provide a simple yet effective linear approximation to solve them. Since many supervised hashing loss formulations can be viewed as energy functions, we can consider our method as a general framework that is able to incorporate many of them.
- We propose a linear-time variant of EM-KSH to tackle large-scale problems. Besides, we reduce the space complexity from the original $\mathcal{O}(nd^2)$ in BSH to $\mathcal{O}(nd)$.
- We have conducted image retrieval experiments on three real-world datasets. Results show that EM-KSH and EM-SPLH outperform state-of-the-art methods. Using the linear time variant of EM-KSH, we can train 64-bit hashing codes on NUS-WIDE in 20 seconds while retaining a state-of-the-art performance.

2. Notations and Problem Definition

2.1. Notations

Lowercase and uppercase boldface letters denote vectors and matrices, respectively. For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, A_{ij} represents the element at the i -th row and j -th column in \mathbf{A} , and \mathbf{A}^T denotes the transpose of \mathbf{A} . \mathbf{A}_i and \mathbf{A}_j denote column vectors formed by the i -th row and the j -th column of \mathbf{A} , respectively. When \mathbf{A} is a square matrix, we let \mathbf{A}^{-1} be the inverse (if exists) of \mathbf{A} , and $\text{diag}(\mathbf{A})$ be a column vector formed by diagonal elements in \mathbf{A} . \mathbf{I} denotes the identity matrix of appropriate size, and $\mathbf{1}$ is a vector or matrix with all ones of appropriate size. $\|\cdot\|_2$ denotes the spectral norm of a matrix while $\|\cdot\|_F$ denotes the Frobenius norm. For a random variable b , $\mathbb{E}[b]$ returns its expectation.

2.2. Problem Definition

Suppose $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_n]^T \in \mathbb{R}^{n \times p}$ is the data matrix, where \mathbf{X}_i is the feature of the i -th data point. In conventional settings [24, 7], the semantic information is given by a pairwise similarity matrix $\mathbf{S} \in \{-1, 0, 1\}^{n \times n}$, where

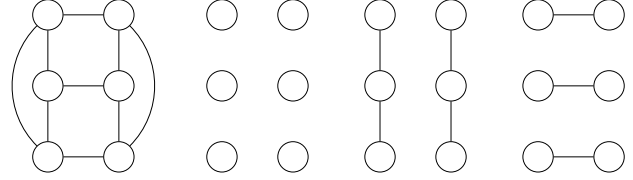


Figure 1. The graphical model of a toy example and its different decomposition structures. From the left to the right are the original structure, the decomposition by element, by column and by row of the original structure, respectively.

$S_{ij} = 1$ denotes that the i -th data point is similar to the j -th, while $S_{ij} = -1$ means they are dissimilar. When $S_{ij} = 0$, we do not know whether they are similar or not. For ease of illustration, we assume that \mathbf{S} is fully observed, that is, $\mathbf{S} \in \{-1, 1\}^{n \times n}$, while our method can tackle the case that partial information is missing naturally. We denote $\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_n]^T = [B_{ik}] \in \{-1, 1\}^{n \times d}$ as the learned hashing matrix, with \mathbf{B}_i as the d -bit hashing code for the i -th data point. The purpose of supervised hashing is to preserve semantic similarities in Hamming space, that is, the Hamming distance between hashing codes of similar data points should be small.

3. Energy Minimizing based Hashing

3.1. General Energy Minimizing Approach

Many supervised hashing loss formulations, for instance, BRE [12], SPLH [20], KSH [16] and LFH [24] can be viewed as exponential losses of polynomials. Inspired by [9, 10], we view a supervised hashing learning formulation as the corresponding densely connected CRF. If we denote a supervised hashing loss as $\mathcal{E}(\mathbf{B}; \mathbf{S})$, the corresponding Gibbs distribution can be written as:

$$p(\mathbf{B}|\mathbf{S}) = \frac{1}{Z} \exp\{-\mathcal{E}(\mathbf{B}; \mathbf{S})\}. \quad (1)$$

To obtain marginal probabilities of this distribution, we must calculate sums over exponentials of energy functions, which is nevertheless intractable. Hence, some form of approximation must be utilized. The mean-field approximation optimizes a distribution $q(\mathbf{B})$ that minimizes the KL divergence $\text{KL}(q||p)$ and factorizes with respect to a partition of variables in \mathbf{B} . Intuitively, the matrix \mathbf{B} has three ways to factorize: by element, by row, and by column. Figure 1 demonstrates a toy graphical model and its three decomposition structures.

Factorizing \mathbf{B} into independent variables provides a tractable way to infer \mathbf{B} . Since each element is a binary random variable, a set of closed-form updating equations can be obtained. The drawback of this approach is that all interaction terms between variables are neglected, so the performance can easily deteriorate.

Factorizing \mathbf{B} by row or by column is more challenge since plenty of interaction terms have to be considered during the optimization process. But if we can obtain the joint distribution of variables by row or by column of \mathbf{B} , then a two-round message passing process could yield more precise marginals than the fully factorized mean-field inference.

Our motivation is to maintain the tractability of the fully factorized distribution while considering interactions between rows or columns of variables in \mathbf{B} . To achieve this goal, we first derive consistency equations from the fully factorized mean-field inference, then approximate a fixed point of these equations. While solving one row or one column of variables, we view others as constants. Solving these equations is intractable since they contain the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$. Therefore, we compute a linear approximation of the sigmoid function on a restricted interval and convert the original consistency equations to a set of linear systems.

To see how the sigmoid function is involved, let us consider minimizing the KL divergence between the fully factorized distribution $q(\mathbf{B}|\Phi) = \prod_i^n \prod_k^d \phi_{ik}^{(B_{ik}+1)/2} (1 - \phi_{ik})^{(1-B_{ik})/2}$ and $p(\mathbf{B}|\mathbf{S})$ in (1), where ϕ_{ik} is the probability that B_{ik} takes the value 1, and $1 - \phi_{ik}$ is the probability that B_{ik} takes the value -1 .

$$\begin{aligned} \text{KL}(q||p) &= \sum_i^n \sum_k^d \phi_{ik} \ln \phi_{ik} + (1 - \phi_{ik}) \ln(1 - \phi_{ik}) \\ &+ \mathbb{E}_q[\mathcal{E}] + \ln Z. \end{aligned} \quad (2)$$

After letting the derivative of ϕ_{ik} be zero, we obtain

$$\phi_{ik} = \sigma\left(-\frac{\partial \mathbb{E}_q[\mathcal{E}]}{\partial \phi_{ik}}\right). \quad (3)$$

Since \mathcal{E} is a polynomial of \mathbf{B} and for a binary random variable b which takes the value 1 with probability ϕ and the value -1 with probability $1 - \phi$, $\mathbb{E}[b] = 1 \times \phi + (-1) \times (1 - \phi) = 2\phi - 1$, (3) usually has the form of

$$\phi = \sigma(\mathbf{A}(2\phi - \mathbf{1}) + \mathbf{b}), \quad (4)$$

where \mathbf{A} is a real symmetric matrix of $d \times d$ and \mathbf{b} is a real vector of $d \times 1$. We wish to solve $\phi \in [0, 1]^{d \times 1}$ approximately.

We restrict the range in which we approximate the sigmoid function to make the approximation error small by defining

$$\lambda = \max_{i \in \{1, \dots, d\}} \left(\sum_j^d |A_{ij}| + |b_i| \right) / c, \quad (5)$$

where $c > 0$ is a constant, and solve the scaled problem

$$\phi = \sigma(\lambda^{-1}(\mathbf{A}(2\phi - \mathbf{1}) + \mathbf{b})). \quad (6)$$

Now each term inside the sigmoid function is bounded by an interval $[-c, c]$. We compute a linear approximation of the sigmoid function: $\sigma(x) \approx c_1 x + c_2$ on this interval, where c_1 and c_2 can be determined by minimizing

$$\min_{c_1, c_2} \int_{-c}^c (\sigma(x) - c_1 x - c_2)^2 dx. \quad (7)$$

Each element in the sigmoid function of (6) is a linear combination of ϕ , so approximating these elements directly may cause considerable error. When \mathbf{A} is invertible, we apply a linear transformation $\mathbf{v} = \lambda^{-1} \mathbf{A}(2\phi - \mathbf{1})$, that is, $2\phi - \mathbf{1} = \lambda \mathbf{A}^{-1} \mathbf{v}$, to make sure that each term in the sigmoid function involves one variable in \mathbf{v} . After this approximation, (6) turns out to be linear equations of \mathbf{v} as

$$(\lambda \mathbf{A}^{-1} - 2c_1 \mathbf{I}) \mathbf{v} = 2c_1 \lambda^{-1} \mathbf{b}, \quad (8)$$

where since $c_2 \equiv 0.5$, it is eliminated automatically.

We mainly discuss the case that \mathbf{A} is invertible, while the case that \mathbf{A} is singular can be treated similarly. In both cases, we have to inverse $(\lambda \mathbf{I} - 2c_1 \mathbf{A})$, so we use the following theorem to ensure its invertibility.

Theorem 1. *A sufficient condition for the invertibility of $(\lambda \mathbf{I} - 2c_1 \mathbf{A})$ is that $2c_1 < 1/c$.*

Proof. We denote the eigenvalue of \mathbf{A} with the largest magnitude as λ_m . According to the Gershgorin circle theorem and $\lambda = \max_{i \in \{1, \dots, d\}} \left(\sum_j^d |A_{ij}| + |b_i| \right) / c$, we have

$$|\lambda_m| \leq \max_{i \in \{1, \dots, d\}} \left(\sum_j^d |A_{ij}| \right) \leq \lambda c. \quad (9)$$

If $2c_1 < 1/c$, the eigenvalue of $2c_1 \mathbf{A}$ with the largest magnitude is less than λ , so $(\lambda \mathbf{I} - 2c_1 \mathbf{A})$ would be positive definite, which concludes the proof. \square

By computing, we find that when $c < 2.5997$, the condition in Theorem 1 holds automatically.

According to whether $\mathbf{b} = \mathbf{0}$, (8) has two cases:

- For $\mathbf{b} \neq \mathbf{0}$, this problem has a closed-form solution

$$\mathbf{v} = 2c_1 \lambda^{-1} (\lambda \mathbf{A}^{-1} - 2c_1 \mathbf{I})^{-1} \mathbf{b}. \quad (10)$$

- For $\mathbf{b} = \mathbf{0}$, this linear system does not have non-zero solution, but we can find the solution of

$$\min_{\mathbf{v}} \left\| (\lambda \mathbf{A}^{-1} - 2c_1 \mathbf{I}) \mathbf{v} \right\|_2^2 \quad (11)$$

instead. The solution of this problem is the eigenvector associated with the smallest eigenvalue of the matrix $(\lambda \mathbf{A}^{-1} - 2c_1 \mathbf{I})^T (\lambda \mathbf{A}^{-1} - 2c_1 \mathbf{I})$.

After solving \mathbf{v} , recall that $\mathbf{v} + \lambda^{-1}\mathbf{b} \triangleq \mathbf{v}' \in [-c, c]$, we first re-normalize \mathbf{v}' by

$$\mathbf{v}' := c \left(2 \left(\frac{\mathbf{v}' - \min(\mathbf{v}')}{\max(\mathbf{v}') - \min(\mathbf{v}')} \right) - 1 \right), \quad (12)$$

then use $\phi = \sigma(\mathbf{v}')$ to obtain the final ϕ .

3.2. Applications to Supervised Hashing

We use two supervised hashing formulations to illustrate how to derive our EM-KSH and EM-SPLH.

For KSH [16], the corresponding Gibbs function is:

$$p(\mathbf{B}|\mathbf{S}) = \exp \left\{ -\frac{1}{4} \sum_{i < j}^n (\mathbf{B}_i^T \mathbf{B}_j - dS_{ij})^2 \right\}. \quad (13)$$

By minimizing the KL divergence between $q(\mathbf{B}|\Phi)$ and $p(\mathbf{B}|\mathbf{S})$, the optimal solution is given by:

$$\begin{aligned} \phi_{ik} &= \sigma \left(-\sum_{j \neq i}^n \sum_{k' \neq k}^d (2\phi_{jk} - 1)(2\phi_{jk'} - 1)(2\phi_{ik'} - 1) \right. \\ &\quad \left. + \sum_{j \neq i}^n dS_{ij}(2\phi_{jk} - 1) \right). \end{aligned} \quad (14)$$

Letting

$$\begin{aligned} A_{kk'}^i &= \sum_{j \neq i}^n -1_{[k \neq k']} (2\phi_{jk} - 1)(2\phi_{jk'} - 1), \\ b_k^i &= \sum_{j \neq i}^n dS_{ij}(2\phi_{jk} - 1), \quad i \in \{1, \dots, n\}, \end{aligned} \quad (15)$$

where $1_{[\cdot]}$ is the indicator function, we recognize that (14) are actually simultaneous equations with the form of (4) over triplets $\{\mathbf{A}^i, \mathbf{b}^i, \Phi_i\}$ where $i \in \{1, \dots, n\}$, so they can be solved using the technique mentioned above.

Another example is SPLH [20]. Its Gibbs distribution can be expressed as:

$$p(\mathbf{B}|\mathbf{S}) = \exp \left\{ \frac{1}{2} \sum_{i < j}^n S_{ij} \mathbf{B}_i^T \mathbf{B}_j \right\}. \quad (16)$$

The corresponding re-estimation equations are given by:

$$\phi_{ik} = \sigma \left(\sum_{j \neq i}^n S_{ij}(2\phi_{jk} - 1) \right). \quad (17)$$

Since $\mathbf{A}^k = \mathbf{S}$ and $\mathbf{b}^k = \mathbf{0}$ hold for all $k \in \{1, \dots, d\}$, these equations can be decoupled into d identical problems over triplets $\{\mathbf{S}, \mathbf{0}, \Phi_k\}$. Therefore, all hashing bits will be the same after solving these linear systems. In addition,

Φ_k is independent of \mathbf{S} , so we can solve a fixed point in one iteration exactly.

It is worth noting that after approximating consistency equations of KSH, the parameter $\mathbf{b} \neq \mathbf{0}$, while for SPLH, $\mathbf{b} \equiv \mathbf{0}$. Consequently, they correspond to two cases of our linear approximation method, respectively. We shall evaluate both of them in the experimental section.

3.3. Stochastic Learning

Due to the unbearable time cost and space complexity for tackling the whole similarity matrix, we follow the same method as LFH [24] to reduce the complexity, i.e., sampling m columns in the original similarity matrix randomly. The resulting sub-matrix is denoted as $\mathbf{S} \in \{-1, 1\}^{n \times m}$. We partition the learned Φ as $\Phi = [\Phi_1, \Phi_2]^T$ where $\Phi_1 \in [0, 1]^{m \times m}$ is the hashing matrix for anterior m points, while $\Phi_2 \in [0, 1]^{(n-m) \times m}$ is the hashing matrix for the later $(n-m)$ data points. We show how to learn semantic information in \mathbf{S} as follows. For $i \in \{1, \dots, m\}$, \mathbf{A}^i and \mathbf{b}^i are exactly the same as in (15). For $i \in \{m+1, \dots, n\}$, we have

$$\begin{aligned} A_{kk'}^i &= \sum_{j=1}^m -1_{[k \neq k']} (2\phi_{jk} - 1)(2\phi_{jk'} - 1), \\ b_k^i &= \sum_{j=1}^m dS_{ij}(2\phi_{jk} - 1), \quad i \in \{m+1, \dots, n\}. \end{aligned} \quad (18)$$

It is interesting to find out that for all $i \in \{m+1, \dots, n\}$, \mathbf{A}^i is the same matrix as \mathbf{A} . This property can be used to reduce the time and space complexity of EM-KSH. To see this, we first write down corresponding linear systems:

$$(\lambda_i \mathbf{A}^{-1} - 2c_1 \mathbf{I}) \mathbf{v}_i = 2c_1 \lambda_i^{-1} \mathbf{b}_i, \quad i \in \{m+1, \dots, n\}. \quad (19)$$

We perform the eigen decomposition of \mathbf{A} as $\mathbf{A} = \mathbf{PDP}^{-1}$ to accelerate the calculation, and solve \mathbf{v}_i as

$$\begin{aligned} \mathbf{v}_i &= 2c_1 \mathbf{P} (\lambda_i \mathbf{D}^{-1} - 2c_1 \mathbf{I})^{-1} \mathbf{P}^{-1} \frac{\mathbf{b}^i}{\lambda_i} \\ &= 2c_1 \mathbf{P} \text{diag} \left((\lambda_i \mathbf{D}^{-1} - 2c_1 \mathbf{I})^{-1} \right) \odot (\mathbf{P}^{-1} \frac{\mathbf{b}^i}{\lambda_i}), \end{aligned} \quad (20)$$

where \odot is the Hadamard product (element-wise product) of two matrices (vectors). Hence, we do not need to spend $\mathcal{O}(d^2(n-m))$ space to store all $(\lambda_i \mathbf{D}^{-1} - 2c_1 \mathbf{I})$, instead we could complete necessary calculations at the cost of $\mathcal{O}(d(n-m) + d^2)$. Besides, Φ_i is independent of \mathbf{A} for $i \in \{m+1, \dots, n\}$, so we can solve Φ_2 in one iteration. Our learning process is summarized in Algorithm 1. In general, $2 \leq T \leq 10$ is enough to get satisfactory performance. We set $T = 3$ throughout our experiments for time-saving purposes.

Algorithm 1: Learning procedure for EM-KSH

Input : $\mathbf{S} \in \{-1, 1\}^{n \times m}$, c, d, T .
Output: $\Phi \in [0, 1]^{n \times d}$, which will be rounded to obtain a hashing matrix $\mathbf{B} \in \{-1, 1\}^{n \times d}$.

Initialize the matrix Φ by randomization.

for $t \leftarrow 1$ **to** T **do**

 Construct $\mathbf{A}^i, \mathbf{b}^i$ according to (15),
 where $i \in \{1, \dots, m\}$.
 Solve the resulting m linear systems using (10),
 and compute Φ_1 .

end

Construct \mathbf{A}, \mathbf{b}^i according to (18),
where $i \in \{m+1, \dots, n\}$.

Compute the eigendecomposition of \mathbf{A} .

Solve the resulting $n - m$ linear systems using (20),
and compute Φ_2 .

return $\Phi = [\Phi_1, \Phi_2]^T$.

3.4. Rounding and Out-of-Sample Extension

We follow the same procedure as in BSH [4] for both rounding and out-of-sample extension. That is, we round Φ according to the mean value of each bit. For out-of-sample extension, we simply learn a linear mapping \mathbf{W} from \mathbf{X} to Φ by minimizing

$$\min_{\mathbf{W}} \|\Phi - \mathbf{X}\mathbf{W}\|_F^2 + \lambda_h \|\mathbf{W}\|_F^2, \quad (21)$$

where λ_h is a regularization hyper-parameter. For a new data point \mathbf{x} , the corresponding ϕ is calculated as

$$\phi = \mathbf{W}^T \mathbf{x}. \quad (22)$$

3.5. Complexity Analysis

We discuss the case that $\mathbf{S} \in \{-1, 1\}^{n \times m}$ is a sub-matrix of the original similarity matrix. For $i \in \{1, \dots, m\}$, we need $\mathcal{O}(m(n-1)d^2 + m(n-1)d) = \mathcal{O}(mnd^2)$ time to calculate all \mathbf{A}^i and \mathbf{b}^i . Then it takes $\mathcal{O}(d^3)$ to solve an equation as in (6), hence, calculating Φ_1 costs $\mathcal{O}(mnd^2 + md^3)$. For $i \in \{m+1, \dots, n\}$, $\mathcal{O}(md^2 + (n-m)md)$ is required to obtain the common \mathbf{A} and all \mathbf{b}^i . Then, by using the trick in (20), we are able to solve one problem like this in $\mathcal{O}(d^2)$ time, and $\mathcal{O}(d^3)$ is needed to compute the eigendecomposition of \mathbf{A} . Since $d \ll m$, updating Φ_2 needs $\mathcal{O}((n-m)md)$. Provided that we have to compute Φ_1 for T times and Φ_2 in one iteration, the total time of computing Φ is bounded by $\mathcal{O}(Tnmd^2)$. Since m is usually chosen as a constant like 1000 and the factor in $\mathcal{O}(\cdot)$ is usually quite small, this method is very fast in real applications.

For the space complexity, $\mathcal{O}(m(d^2 + d))$ is occupied by \mathbf{A}^i and \mathbf{b}^i for $i \in \{1, \dots, m\}$. By utilizing the trick in

(20), it only takes $\mathcal{O}((n-m)d + d^2)$ space for learning Φ_2 . Therefore, the total storage cost is $\mathcal{O}(md^2 + nd)$. In most cases, we have $md = \mathcal{O}(n)$, hence the space complexity can be written as $\mathcal{O}(nd)$, which outperforms the original $\mathcal{O}(nd^2)$ in BSH [4].

3.6. Extensions of Other Methods

Extending our method to BRE [12] and ExpH [14] is quite straightforward. Here we only show how to extend our proposed technique to LFH [24]. The optimization objective of LFH is

$$p(\mathbf{B}|\mathbf{S}) = \prod_{i < j} \sigma(\mathbf{B}_i^T \mathbf{B}_j)^{\frac{1+S_{ij}}{2}} (1 - \sigma(\mathbf{B}_i^T \mathbf{B}_j))^{\frac{1-S_{ij}}{2}}, \quad (23)$$

which is not a Gibbs distribution at first glance. But after applying the local variational method in [6], the original objective is lower bounded by

$$\tilde{p}(\mathbf{B}|\mathbf{S}) = \frac{1}{Z} \exp \left\{ \sum_{i < j} \left(\frac{1}{2} S_{ij} \mathbf{B}_i^T \mathbf{B}_j + \lambda(\xi_{ij}) (\mathbf{B}_i^T \mathbf{B}_j)^2 \right) \right\}, \quad (24)$$

where $\xi_{ij} = \sqrt{\mathbb{E}[\mathbf{B}_i^T \mathbf{B}_j]^2}$, and $\lambda(\xi_{ij}) = -\frac{\sigma(\xi_{ij}) - \frac{1}{2}}{2\xi_{ij}}$. The resulting mean-field consistency equations are given by

$$\phi_{ik} = \sigma \left(\sum_{j \neq i} \sum_{k' \neq k} (4\lambda(\xi_{ij})(2\phi_{jk} - 1)(2\phi_{jk'} - 1) \right. \\ \left. (2\phi_{ik'} - 1) + \sum_{j \neq i} S_{ij}(2\phi_{jk} - 1) \right). \quad (25)$$

So we can solve these consistency equations by finding a fixed point in the same manner as EM-KSH.

Interestingly, we have found a deep connection between KSH and LFH. Suppose that for arbitrary two codes \mathbf{B}_i and \mathbf{B}_j , we have $\mathbb{E}[\mathbf{B}_i] = \mathbb{E}[\mathbf{B}_j]$ or $\mathbb{E}[\mathbf{B}_i] = -\mathbb{E}[\mathbf{B}_j]$, then $\xi_{ij} = d$ and $\lambda(\xi_{ij}) \approx -\frac{1}{4d}$. Substituting $\lambda(\xi_{ij}) \approx -\frac{1}{4d}$ into (25) yields

$$\phi_{ik} = \sigma \left(- \sum_{j \neq i} \sum_{k' \neq k} \frac{1}{d} ((2\phi_{jk} - 1)(2\phi_{jk'} - 1) \right. \\ \left. (2\phi_{ik'} - 1) + \sum_{j \neq i} S_{ij}(2\phi_{jk} - 1) \right), \quad (26)$$

which is identical to (14) except for a factor of d . Consequently, KSH can be viewed as a hard assignment of hashing codes to -1 or 1 , while LFH makes a soft assignment based on probabilities.

Method	ESPGAME				CIFAR-10			
	8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
CCA-ITQ	0.2751	0.2805	0.2804	0.2816	0.2163	0.2215	0.0.2254	0.2319
KSH	0.2977	0.3086	0.3194	0.3248	0.2521	0.2825	0.3210	0.3492
LFH	0.3116	0.3330	0.3546	0.3659	0.2881	0.3996	0.5216	0.6085
SDH	0.3094	0.3290	0.3312	0.3388	0.3329	0.4833	0.5397	0.5865
COSDISH	0.2977	0.3158	0.3327	0.3407	0.4898	0.5733	0.6215	0.6369
NSH	0.2946	0.3058	0.3116	0.3223	0.3907	0.4476	0.4875	0.5298
BSH	0.3314	0.3456	0.3583	0.3639	0.4132	0.4989	0.5792	0.6137
EM-KSH	0.3430	0.3602	0.3592	0.3710	0.4459	0.5344	0.5804	0.6276

Table 1. Experimental performance on ESPGAME and CIFAR-10 in terms of mAP. Best results are in bold.

Code length	8 bits		16 bits		32 bits		64 bits	
	mAP	Time	mAP	Time	mAP	Time	mAP	Time
CCA-ITQ	0.2947	4.38	0.3002	4.65	0.3111	6.02	0.3152	13.76
KSH	0.3732	211.44	0.4149	732.87	0.4356	1792.90	0.4391	2931.97
LFH	0.4542	31.05	0.4854	54.50	0.5120	82.29	0.5324	138.72
SDH	0.4353	46.14	0.4630	56.59	0.4847	155.17	0.5143	649.32
COSDISH	0.4299	18.71	0.4827	28.07	0.4918	276.28	0.5231	1030.22
NSH	0.3837	17.72	0.4197	19.32	0.4438	24.51	0.4385	31.95
BSH	0.4683	15.19	0.4825	20.85	0.5076	39.17	0.5220	121.24
EM-KSH	0.4878	10.36	0.5120	10.53	0.5331	12.12	0.5434	17.52

Table 2. The mAP and the corresponding training time (in seconds) on NUS-WIDE. Best results are in bold.

4. Experiments

4.1. Datasets

We evaluate our proposed method on three image datasets: NUS-WIDE¹ [1], CIFAR-10² [11] and ESPGAME³. All of them have been widely used for supervised hashing methods evaluation [18, 7, 4].

CIFAR-10 consists of 60,000 color images which are manually categorized into 10 classes. Each image in this dataset is represented by a 512-dimensional GIST feature vector. Two images are considered to be similar if they are of the same class, otherwise, they are treated as dissimilar.

The ESPGAME dataset contains 20,770 images with 268 keywords while the NUS-WIDE dataset includes 269,648 natural images with 81 tags. During experiments, we use 512-dimensional GIST features and 500-dimensional bag-of-words features for ESPGAME and NUS-WIDE, respectively. For these two datasets, two images are considered as semantic neighbors if they share at least one common tag.

¹<http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm>

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

³<http://www.hunch.net/~jl/>

4.2. Experimental Settings

Following [24, 7], for all datasets, we randomly select 1000 data points as the validation set and 1000 points as the query set. In the preprocessing phase, we perform normalization on features to make each dimension have zero mean and same variance. The default value of c is 2, so the linear approximation of the sigmoid function is $\sigma(x) \approx 0.2109x + 0.5$ on $[-2, 2]$, while the default value of m is 1000. Since our method directly solves a fixed point of consistency mappings, very few iterations are already enough to obtain satisfactory results. For all experiments, we set $T = 3$.

Since supervised hashing methods outperform unsupervised ones in preserving semantic similarities, we simply compare our proposed method with other state-of-the-art supervised hashing methods, including CCA-ITQ [3], KSH [16], LFH [24], SDH [18], COSDISH [7], NSH [15] and BSH [4]. The code of all these methods is implemented by corresponding authors. For all methods, we follow settings the same as those suggested by these authors. All our experiments are conducted on a workstation with 16 Intel i7-6900K CPU cores and 64GB RAM, and all the results are the average value of 10 random partitions.

For these three datasets, we report the compared results in terms of Hamming ranking. For each query, all the data points in the training set are sorted ascending according to

the Hamming distance between their hashing codes and the code of the query. The mean average precision (mAP) is used to evaluate different methods.

4.3. Comparing EM-KSH with baselines

Table 1 and 2 show the mAP of our proposed EM-KSH and other methods on these three datasets. In addition, the training time in seconds on the NUS-WIDE dataset with various code lengths is reported in Table 2. By comparing our EM-KSH with other baselines, we discover that for ESPGAME and NUS-WIDE, our method outperforms other baselines in almost all cases. Especially, EM-KSH achieves much better performance than the original KSH for all these three datasets consistently, which justifies that the fixed point solved by EM-KSH is quite desirable. For CIFAR-10, as we can see, COSDISH is the most effective hashing method. Recall that for a single-label dataset, the equality relation over the set of labels is transitive, that is, for three labels l_i, l_j, l_k , if $l_i = l_j$ and $l_j = l_k$, then $l_i = l_k$. We speculate that the error bound of the $2d$ -approximation algorithm being used in COSDISH would tighten when the transitive relation exists in the similarity matrix.

For the time complexity, the performance of our method is even more superior. As we can see, only CCA-ITQ can be slightly faster than EM-KSH, but our method outperforms CCA-ITQ by a large margin. Besides, EM-KSH is orders of magnitude faster than other state-of-the-art methods, especially for learning 64-bit hashing codes on NUS-WIDE. In fact, the time spent for EM-KSH to learn 64-bit codes on NUS-WIDE is less than or equal to the time that other baselines used to learn 8-bit codes.

In summary, our EM-KSH yields the best performance on two datasets with multiple tags and is almost as fast as CCA-ITQ. Although COSDISH outperforms EM-KSH on the CIFAR-10 dataset, we still argue that our method is state-of-the-art. First, multi-label images are easier to collect and have more real-world applications. Second, EM-KSH is much faster than COSDISH, so we can train longer hashing codes to defeat COSDISH on single-label datasets with a reasonable time cost. In addition, our method can be easily modified to accommodate different hashing formulations, while COSDISH is less flexible.

4.4. Sensitivity to Hyper-parameters

We vary the hyper-parameter c from 0.5 to 2.5, and report the mAP performance on CIFAR-10 and NUS-WIDE in Figure 2. We find that EM-KSH is not sensitive to the value of c and can achieve good performance consistently.

In Figure 3, we show a performance comparison of EM-KSH and other baselines with the size of the query set q and the number of columns in the similarity sub-matrix m take values from 1000, 2000, 3000, 5000 and 10000 on NUS-WIDE. In almost all cases, the performance of EM-KSH

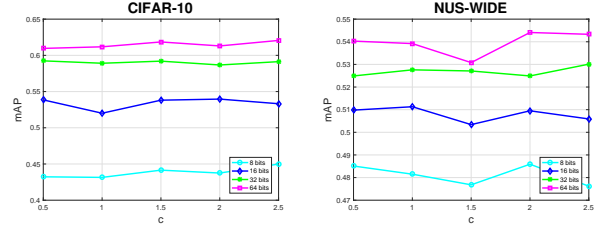


Figure 2. Sensitivity to hyper-parameter c .

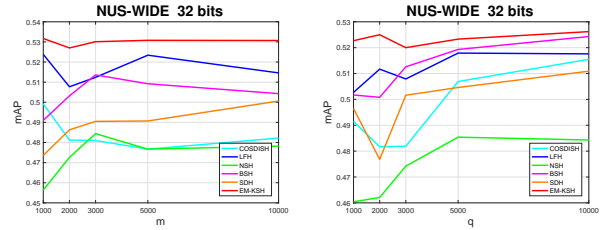


Figure 3. Sensitivity to hyper-parameters m and q .

is superior to other baselines.

4.5. Long Hashing Codes Learning

In many applications, space is not the primary bottleneck, so longer hashing codes might be required to further boost the retrieval precision. We train 128-bit and 256-bit codes using EM-KSH and other baselines on NUS-WIDE and show the result in Table 3 to further demonstrate the effectiveness of our method.

Code length	128 bits		256 bits	
	mAP	Time	mAP	Time
COSDISH	0.5318	691.73	0.5405	3096.79
LFH	0.5378	71.24	0.5548	135.23
NSH	0.4415	22.82	0.4606	43.94
BSH	0.5285	508.64	OOM	OOM
EM-KSH	0.5509	27.12	0.5615	55.68

Table 3. The mAP and corresponding training time in seconds on NUS-WIDE for learning 128-bit and 256-bit codes. Best results are in bold. OOM means out-of-memory error.

During experiments, BSH runs out of memory and terminates while learning 256-bit codes, while our method can easily learn codes up to 256 bits with a rather low time cost. The results further indicate our improvement on the space side is significant.

4.6. Comparing EM-SPLH with baselines

EM-KSH corresponds to the case $\mathbf{b} \neq \mathbf{0}$ for our proposed linear approximation method, so we also report the performance of EM-SPLH corresponding to the case $\mathbf{b} = \mathbf{0}$

for completeness. Since there is no interaction term in the formulation of SPLH, all bits learned by finding a fixed point should be the same. Consequently, we evaluate our EM-SPLH and other state-of-the-art baselines by just learning 1-bit code. For all three datasets, we randomly choose 1000 points as the training set and another 1000 as the test set. The results are the mean of 10 independent partitions.

Method	ESPGAME	CIFAR-10	NUS-WIDE
LFH	0.4890	0.1619	0.3740
COSDISH	0.4773	0.1054	0.4065
NSH	0.4957	0.1714	0.4646
BSH	0.5009	0.1409	0.3893
EM-KSH	0.4953	0.1558	0.4703
EM-SPLH	0.5110	0.1573	0.5015

Table 4. The mAP performance of learning 1-bit hashing code on three datasets. Best results are in bold.

As shown in Table 4, EM-SPLH outperforms several state-of-the-art methods on ESPGAME and NUS-WIDE datasets, including EM-KSH. For CIFAR-10, the performance of our method is also competitive. In fact, for the 1-bit case, formulations of KSH and SPLH are equivalent since $(dS_{ij} - b_i b_j)^2 = -2dS_{ij}b_i b_j + const$ holds for $d = 1$. Besides, LFH, COSDISH, BSH and EM-KSH can all be viewed as optimizing the formulation of KSH (COSDISH and EM-KSH optimize the original KSH, while LFH and BSH optimize the KSH with soft assignments). By comparing the results, we can find the fixed point solved by our method is quite desirable.

5. Conclusion

In this paper, we have proposed a novel method to approximate a fixed point of consistency mappings deriving from mean-field inference. We convert these consistency equations to linear systems by a linear approximation of the sigmoid function to obtain a closed-form solution. By using this technique to supervised hashing problem, we obtain EM-KSH and EM-SPLH. Experimental results on three image datasets show that our methods outperform other state-of-the-art methods.

References

- [1] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., 2009.
- [2] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.
- [3] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer*

- Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 817–824. IEEE, 2011.
- [4] Z. Hu, J. Chen, H. Lu, and T. Zhang. Bayesian supervised hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [5] G. Irie, Z. Li, X.-M. Wu, and S.-F. Chang. Locally linear hashing for extracting non-linear manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2115–2122, 2014.
- [6] T. S. Jaakkola and M. I. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10(1):25–37, 2000.
- [7] W.-C. Kang, W.-J. Li, and Z.-H. Zhou. Column sampling based discrete supervised hashing. In *AAAI*, 2016.
- [8] W. Kong and W.-J. Li. Isotropic hashing. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2012.
- [9] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.
- [10] P. Krähenbühl and V. Koltun. Parameter learning and convergent inference for dense random fields. In *International Conference on Machine Learning*, pages 513–521, 2013.
- [11] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- [12] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in neural information processing systems*, pages 1042–1050, 2009.
- [13] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1963–1970, 2014.
- [14] G. Lin, C. Shen, D. Suter, and A. van den Hengel. A general two-step approach to learning-based hashing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2552–2559, 2013.
- [15] Q. Liu and H. Lu. Natural supervised hashing. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1788–1794, 2016.
- [16] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2074–2081. IEEE, 2012.
- [17] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 353–360, 2011.
- [18] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–45, 2015.
- [19] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010.

- [20] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 1127–1134, 2010.
- [21] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang. Learning hash codes with listwise supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3032–3039, 2013.
- [22] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.
- [23] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *International conference on machine learning*, volume 6, page 7, 2014.
- [24] P. Zhang, W. Zhang, W.-J. Li, and M. Guo. Supervised hashing with latent factor models. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 173–182. ACM, 2014.