Fast Structural Binary Coding

Dongjin Song*, Wei Liu♯, and David A. Meyer†
*Department of Electrical and Computer Engineering,University of California, San Diego
La Jolla, USA, 92093-0409. Email: dosong@ucsd.edu
♯ Didi Research, Didi Kuaidi, Beijing, China. Email: wliu@ee.columbia.edu
†Department of Mathematics,University of California, San Diego
La Jolla, USA, 92093-0112. Email: dmeyer@math.ucsd.edu

Abstract

Binary coding techniques, which compress originally high-dimensional data samples into short binary codes, are becoming increasingly popular due to their efficiency for information retrieval. Leveraging supervised information can dramatically enhance the coding quality, and hence improve search performance. There are few methods, however, that efficiently learn coding functions that optimize the precision at the top of the Hamming distance ranking list while approximately preserving the geometric relationships between database examples. In this paper, we propose a novel supervised binary coding approach, namely Fast Structural Binary Coding (FSBC), to optimize the precision at the top of a Hamming distance ranking list and ensure that similar images can be returned as a whole. The key idea is to train disciplined coding functions by optimizing a lower bound of the area under the ROC (Receiver Operating Characteristic) curve (AUC) and penalize this objective so that the geometric relationships between database examples in the original Euclidean space are approximately preserved in the Hamming space. To find such a coding function, we relax the original discrete optimization objective with a continuous surrogate, and then derive a stochastic gradient descent method to optimize the surrogate objective efficiently. Empirical studies based upon two image datasets demonstrate that the proposed binary coding approaches achieve superior image search performance to the states-of-the-art.

1 Introduction

With the rapid development of massive image collection applications such as Instagram, Flickr, and Pinterest, there is an increasing demand for finding visually relevant images effectively and efficiently. Binary coding techniques, rather than exhaustively searching for the most similar images with respect to a query in a high-dimensional feature space, encode images with compact binary codes and conduct efficient searches in the generated low-dimensional code space (*i.e.*, Hamming space). This can reduce search time and save storage space.

In particular, binary coding methods aim to learn a set of coding functions $\left\{h_q : \mathbb{R}^d \mapsto \mathbb{H} = \{-1, 1\}\right\}_{q=1}^{r}$ to map data samples from a d-dimensional data space $\mathbb{R}^d$ to an r-dimensional Hamming space $\mathbb{H}^r$. Early binary coding approaches, *e.g.*, Locality-Sensitive Hashing (LSH) [Andoni and Indyk, 2008] and Min-wise Hashing (MinHash) [Broder *et al.*, 1998], produce binary codes with random permutations or projections. These randomized binary coding methods, however, require long code lengths ($r \geq 1,000$) to meet search requirements, and usually cannot perform well for large-scale image search [Liu *et al.*, 2012; 2014; Zhang *et al.*, 2014; Wang *et al.*, 2014; 2016] as they consider data points independently.

In contrast to such randomized binary coding approaches, various *data-dependent* binary coding methods have been invented more recently. These techniques, in general, can be divided into two main categories: unsupervised and supervised (including semi-supervised) approaches. Unsupervised approaches, such as Spectral Hashing (SH) [Weiss *et al.*, 2008], Iterative Quantization (ITQ) [Gong *et al.*, 2012], Isotropic Hashing (ISOH) [Kong and Li, 2012], Discrete Graph Hashing (DGH) [Liu *et al.*, 2014] *etc.*, learn coding functions by modeling the underlying data structures, distributions, or topological information. Supervised approaches, on the contrary, learn coding functions by leveraging supervision information, *e.g.*, instance-level labels, pair-level labels, or triplet-level ranks. Representative techniques include pointwise supervised methods (*e.g.*, Binary Reconstructive Embedding (BRE) [Kulis and Darrell, 2009]), pairwise supervised methods (*e.g.*, Minimal Loss Hashing (MLH) [Norouzi and Fleet, 2011] and Kernel-based Supervised Hashing (KSH) [Liu *et al.*, 2012]); and rank supervised approaches (*e.g.*, Hamming Distance Metric Learning (HDML) [Norouzi *et al.*, 2012], Ranking-based Supervised Hashing (RSH) [Wang *et al.*, 2013], Column Generation Hashing (CGH) [Li *et al.*, 2013]), Rank Preserving Hashing (RPH) [Song *et al.*, 2015b], and Top Rank Supervised Binary Coding (Top-RSBC) [Song *et al.*, 2015a].

Despite existing rank supervised binary coding techniques having shown their effectiveness and efficiency for scalable visual search tasks, few of them focus on optimizing the pre-

cision at the top of a ranking list according to Hamming distance, while considering the underlying structure of the ranking list [Weston and Blitzer, 2012] appropriately (*i.e.*, approximately preserving the geometric relationship between database examples such that the set of similar images can be returned as a whole). Therefore, in this paper, we propose a novel supervised binary coding approach, namely Fast Structural Binary Coding (FSBC), to optimize the precision at the top of a Hamming distance ranking list and approximately preserve the geometric relationships between database examples. The core idea is to train disciplined coding functions by optimizing a lower bound of the area under the ROC (Receiver Operating Characteristic) curve (AUC) and penalizing the objective such that the geometric relationships between database examples in the original Euclidean space is preserved in the Hamming space. In this way, we may avoid producing a set of top ranked images which may be too diverse and include irrelevant examples (*e.g.*, examples from different classes). Since the objective we introduce is discrete and the associated optimization problem is combinatorially difficult, we relax the original discrete objective to a continuous and differentiable surrogate, and then derive a stochastic gradient descent method to optimize the surrogate objective. We compare the proposed approach, FSBC, against various state-of-the-art binary coding techniques through extensive experiments conducted on two benchmark image datasets, *i.e.*, SUN397 [Xiao *et al.*, 2010] and YouTube Faces [Wolf *et al.*, 2011]. The experimental results demonstrate that FSBC outperforms the state-of-the-art approaches for various image search tasks.

2 Lower Bound of AUC

In this section, we first introduce the notation used in the paper. Then we introduce the concept of binary coding. Finally, we derive a lower bound of the area under the ROC (Receiver Operating Characteristic) curve (AUC) based upon binary codes.

2.1 Notation

Let $\mathbf{X} \in \mathbb{R}^{d\times n}$ be a data matrix of n data samples with d dimensions; we can use $\boldsymbol{x}_j \in \mathbb{R}^d$ to represent the j-th column of $\mathbf{X}$, and X_{ij} to denote the entry in i-th row and j-th column of $\mathbf{X}$, respectively. Moreover, we use $\|\cdot\|_{\mathrm{F}}$ to denote the Frobenius norm of matrices, and $\|\boldsymbol{x}\|_{\mathrm{H}}$ to represent the Hamming norm of vector $\boldsymbol{x}$, which is defined as the number of nonzero entries in $\boldsymbol{x}$, *i.e.*, the ℓ_0 norm. We use $\|\boldsymbol{x}\|_1$ to represent the ℓ_1 norm of vector $\boldsymbol{x}$, which is defined as the sum of absolute values of the entries in $\boldsymbol{x}$.

2.2 Binary coding

Given a data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, we aim to learn a set of mapping functions $\big\{h_q(\boldsymbol{x})\big\}_{q=1}^{r}$ such that a d-dimensional floating-point input $\boldsymbol{x} \in \mathbb{R}^d$ is compressed into an r-bit binary code $\boldsymbol{b} = \big[h_1(\boldsymbol{x}), \ldots, h_r(\boldsymbol{x})\big] \in \mathbb{H}^r \equiv \{1, -1\}^r$. This mapping, also called a coding function in the literature, is defined as:

$$h_q(\boldsymbol{x}) = \mathrm{sgn}\big(f_q(\boldsymbol{x})\big), \quad q = 1, \ldots, r, \tag{1}$$

where $\mathrm{sgn}(x)$ is the sign function that returns 1 if $x > 0$ and -1 otherwise, and $f_q : \mathbb{R}^d \mapsto \mathbb{R}$ is a proper prediction function. A variety of mathematical forms for f_q (*e.g.*, linear or nonlinear) can be taken to apply for domain specific practical applications. In this work, we consider a linear prediction function, *i.e.*, $f_q(\boldsymbol{x}) = \boldsymbol{w}_q^\top \boldsymbol{x} + t_q$ (where $\boldsymbol{w}_q \in \mathbb{R}^d$ and $t_q \in \mathbb{R}$) for simplicity. Based upon the previous work [Gong *et al.*, 2012; Kong and Li, 2012; Liu *et al.*, 2012; Wang *et al.*, 2013], we set the bias term $t_q = -\boldsymbol{w}_q^\top \boldsymbol{u}$ by using the mean vector $\boldsymbol{u} = \sum_{i=1}^{n} \boldsymbol{x}_i/n$, which will make each generated binary bit $\big\{h_q(\boldsymbol{x}_i)\big\}_{i=1}^{n}$ for $q \in [1 : r]$ be nearly balanced and hence have maximum entropy. For brevity, we can define a coding function $\boldsymbol{h} : \mathbb{R}^d \mapsto \mathbb{H}^r$ to comprise the functionality of r hash functions $\{h_q\}_{q=1}^{r}$, that is,

$$\boldsymbol{h}(\boldsymbol{x}, \mathbf{W}) = \mathrm{sgn}\big(\mathbf{W}^\top(\boldsymbol{x} - \boldsymbol{u})\big), \tag{2}$$

which is parameterized by a matrix $\mathbf{W} = [\boldsymbol{w}_1 \cdots \boldsymbol{w}_r] \in \mathbb{R}^{d\times r}$. Note that Eq. 2 applies the sign function element-wise. For convenience we write $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x}, \mathbf{W})$.

2.3 Lower bound of AUC

Given a triplet ($\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_s$), assuming $\boldsymbol{x}_i$ is a query, $\boldsymbol{x}_j \in \mathcal{P}$ is a similar example to $\boldsymbol{x}_i$ ($\mathcal{P}$ is the set of similar examples), and $\boldsymbol{x}_s \in \mathcal{N}$ is a dissimilar example to $\boldsymbol{x}_i$ ($\mathcal{N}$ is the set of dissimilar examples), then the AUC for query $\boldsymbol{x}_i$ is given as:

$$AUC = \frac{1}{|\mathcal{P}||\mathcal{N}|}\Big(\sum_{\boldsymbol{x}_j\in\mathcal{P}}\sum_{\boldsymbol{x}_s\in\mathcal{N}} I\Big(\|\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x}_j)\|_{\mathrm{H}} < \|\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x}_s)\|_{\mathrm{H}}\Big)\Big) \tag{3}$$

where $I(\cdot)$ is an indicator function which is 1 if the condition in the parenthesis is satisfied and 0 otherwise.

Since AUC counts each pairwise comparison equally, it does not explicitly quantify the fraction of positive examples which achieve the optimal ranking (ranked on the top of a Hamming distance ranking list). For this purpose, we derive a lower bound of AUC in Theorem 1.

Theorem 1. *AUC is lower bounded by*

$$AUC \geq \frac{1}{|\mathcal{P}|}\Big(\sum_{\boldsymbol{x}_j\in\mathcal{P}}\prod_{\boldsymbol{x}_s\in\mathcal{N}} I\Big(\|\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x}_j)\|_{\mathrm{H}} < \|\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x}_s)\|_{\mathrm{H}}\Big)\Big), \tag{4}$$

with equality holding if within each product operator the condition for each indicator function is jointly satisfied or jointly not satisfied.

Theorem 1 states that the fraction of positive examples which achieve the optimal ranking cannot be greater than AUC. It can be proved using the fact that the arithmetic mean is always greater than or equal to the geometric mean. Note that the calculation of the AUC lower bound in Eq. 4 includes all the possible pairwise comparisons, which may make the underlying optimization problem intractable for a large scale database. An equivalent, more tractable form, can be derived:

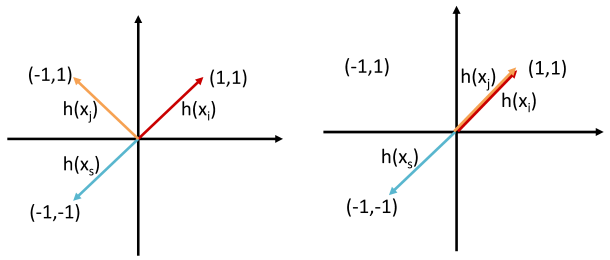


Figure 1: Given a triplet ($h(\boldsymbol{x}_i)$,$h(\boldsymbol{x}_j)$,$h(\boldsymbol{x}_s)$) (a) If $\lambda = 0$, $h(\boldsymbol{x}_i) = (1,1)$, $h(\boldsymbol{x}_j) = (-1,1)$, and $h(\boldsymbol{x}_s) = (-1,-1)$ could be one of the solutions for Eq. 6. (b) If $\lambda > 0$, $h(\boldsymbol{x}_i) = (1,1)$, $h(\boldsymbol{x}_j) = (1,1)$, and $h(\boldsymbol{x}_s) = (-1,-1)$ will be the optimal solution.

Proposition 1. *The lower bound of AUC in Theorem 1 is equivalent to*

$$\frac{1}{|\mathcal{P}|}\Big(\sum_{\boldsymbol{x}_j\in\mathcal{P}} I\Big(\|\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{h}(\boldsymbol{x}_j)\|_{\mathrm{H}} < \min_{\boldsymbol{x}_s\in\mathcal{N}}\big(\|\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{h}(\boldsymbol{x}_s)\|_{\mathrm{H}}\big)\Big)\Big), \tag{5}$$

which can be calculated in linear time.

Proposition 1 suggests that instead of exhaustively searching for the pairwise comparisons which are jointly satisfied, we only need to compare with the minimum Hamming distance over the set $\mathcal{N}$. Proposition 1 also quantifies the fraction of positive examples which are ranked on top of all the negative examples (*i.e.*, on top of the ranking list) [Song *et al.*, 2015c].

3 Fast Structural Binary Coding

In this section, we first present Fast Structural Binary Coding (FSBC). Then we derive a stochastic gradient method to produce binary codes by optimizing the FSBC objective.

3.1 Model

FSBC aims to optimize the precision at the top of a Hamming distance ranking list and approximately preserve the geometric relationship between database examples based upon a linear mapping $\mathbf{W}$. Given a triplet ($\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_s$), assuming $\boldsymbol{x}_i$ is a query, $\boldsymbol{x}_j$ is a similar example, and $\boldsymbol{x}_s = \arg\min_{\boldsymbol{x}\in\mathcal{N}} \|\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{h}(\boldsymbol{x})\|_{\mathrm{H}}$ is a dissimilar example which is closest to $\boldsymbol{x}_i$ in Hamming space within the set $\mathcal{N}$, the objective of FSBC can be given as:

$$\begin{aligned}\mathcal{O}(\mathbf{W}) = {} & I\Big(\|\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{h}(\boldsymbol{x}_j)\|_{\mathrm{H}} < \|\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{h}(\boldsymbol{x}_s)\|_{\mathrm{H}}\Big) \\ & + \frac{\lambda}{2}\Big(\|\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{h}(\boldsymbol{x}_s)\|_2^2 + \|\boldsymbol{h}(\boldsymbol{x}_j)-\boldsymbol{h}(\boldsymbol{x}_s)\|_2^2 \\ & - \|\boldsymbol{h}(\boldsymbol{x}_i)-\boldsymbol{h}(\boldsymbol{x}_j)\|_2^2\Big) - \frac{\mu}{2}\|\boldsymbol{W}\|_F^2,\end{aligned} \tag{6}$$

where the first term encodes the lower bound of AUC in Proposition 1. The second term approximately preserves the geometric relationship of the triplet in the original Euclidean space (*i.e.*, the distance of dissimilar pairs $(\boldsymbol{h}(\boldsymbol{x}_j), \boldsymbol{h}(\boldsymbol{x}_s))$ and $(\boldsymbol{h}(\boldsymbol{x}_i), \boldsymbol{h}(\boldsymbol{x}_s))$ should be large, and the distance of similar pair $(\boldsymbol{h}(\boldsymbol{x}_i), \boldsymbol{h}(\boldsymbol{x}_j))$ should be small). This term is necessary because (1) it resolves the degeneracy problem shown in Figure 1; (2) it imposes a constraint over the dissimilar pair $(\boldsymbol{h}(\boldsymbol{x}_j), \boldsymbol{h}(\boldsymbol{x}_s))$ to ensure the distance between them is also large; (3) it also explicitly states that the distances between $(\boldsymbol{h}(\boldsymbol{x}_i), \boldsymbol{h}(\boldsymbol{x}_s))$ and between $(\boldsymbol{h}(\boldsymbol{x}_i), \boldsymbol{h}(\boldsymbol{x}_j))$ should be large/small respectively, while the first term only cares about their difference. The last term is a regularization term to prevent the model from overfitting. $\lambda > 0$ and $\mu > 0$ are two hyper-parameters to control the balance of the three terms.

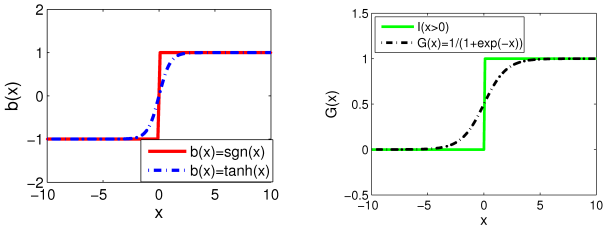


Figure 2: Relaxation of the objective function. (a) $\tanh(x)$ is a relaxation of $\mathrm{sgn}(x)$; (b) sigmoid loss $G(x) = \frac{1}{1+\exp(-x)}$ is a good approximation for indicator function $I(x > 0)$.

3.2 Relaxation and approximation

The proposed model in Eq. 6 is difficult to optimize because (1) the coding function in Eq. 2 is a discrete mapping; (2) the Hamming norm lies in a discrete space; and (3) the indicator function in Eq. 6 is non-differentiable. Thus the objective in Eq. 6 is discrete, and combinatorially difficult to optimize.

To address these issues we relax the original discrete objective to a continuous and differentiable surrogate. We first approximate the original coding function $\boldsymbol{h}(\boldsymbol{x}, \mathbf{W}) = \mathrm{sgn}\big(\mathbf{W}^\top(\boldsymbol{x}-\boldsymbol{u})\big)$ by

$$\overline{\boldsymbol{h}}(\boldsymbol{x}, \mathbf{W}) = \tanh\big(\mathbf{W}^\top(\boldsymbol{x}-\boldsymbol{u})\big), \tag{7}$$

which is continuous and differentiable as shown in Figure 2(a). For convenience we write $\overline{\boldsymbol{h}}(\boldsymbol{x}_i) = \overline{\boldsymbol{h}}(\boldsymbol{x}_i, \mathbf{W})$.

Second, we relax the Hamming norm in Eq. 6 to the ℓ_1 norm which is convex and robust to outliers. Finally, we approximate the indicator function in Eq. 6 with the sigmoid function, $G(x) = \frac{1}{1+\exp(-x)}$, as shown in Figure 2(b), *i.e.*,

$$\begin{aligned} & I\Big(\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_j)\|_1 < \|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_s)\|_1\Big) \\ \approx {} & G\Big(\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_s)\|_1 - \|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_j)\|_1\Big).\end{aligned} \tag{8}$$

The basic idea is that if $\overline{\boldsymbol{h}}(\boldsymbol{x}_i)$ is closer to $\overline{\boldsymbol{h}}(\boldsymbol{x}_j)$ than $\overline{\boldsymbol{h}}(\boldsymbol{x}_s)$ in the ℓ_1 norm, then the value of this objective should be close to 1.

With these relaxations/approximations, the original objec-

Algorithm 1 Fast Structural Binary Coding

Input: $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s\}$, α, $\mathbf{W}$, λ, and μ
Output: $\mathbf{W} \in \mathbb{R}^{d\times k}$
Repeat
  Randomly pick up a sample $\boldsymbol{x}_i$.
  Fix $\boldsymbol{x}_i$ and randomly select a similar sample $\boldsymbol{x}_j$.
  Fix $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, randomly draw p dissimilar sample $\boldsymbol{x}_s$ when s varies to form $\{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s\}_{s=1}^{p}$.
  Determine s by $\min_{s\in(1,\ldots,p)} \big(\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\|_1\big)$.
  If $\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\|_1 < \epsilon + \|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j)\|_1$
    Calculate $\frac{\partial\overline{\mathcal{O}}(\mathbf{W})}{\partial\mathbf{W}}$ based upon Eq. 10, 11, 12.
    Make a gradient ascent based upon Eq. 13.
  End if
Until mean average precision does not improve or maximum iteration number is achieved.

tive in Eq. 6 can be fomulated as:

$$\begin{aligned}\overline{\mathcal{O}}(\mathbf{W}) = G\Big(\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\|_1 - \|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j)\|_1\Big) \\ + \frac{\lambda}{2}\Big(\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\|_2^2 + \|\overline{\boldsymbol{h}}(\boldsymbol{x}_j) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\|_2^2 \\ - \|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j)\|_2^2\Big) - \frac{\mu}{2}\|\mathbf{W}\|_F^2,\end{aligned} \tag{9}$$

where s is the index of the dissimilar example closest to $\boldsymbol{x}_i$ in Hamming space, deefined by $\boldsymbol{x}_s = \arg\min_{\boldsymbol{x}\in\mathcal{N}} \|\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x})\|_1$.

3.3 Optimization

To optimize the approximated objective in Eq. 9, in each iteration we first randomly select a query $\boldsymbol{x}_i$, a similar example $\boldsymbol{x}_j$, and a set $\mathcal{N}$ of p dissimilar examples. After determining $\boldsymbol{x}_s$ as $\arg\min_{\boldsymbol{x}\in\mathcal{N}} \big(\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x})\|_1\big)$, then if $\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\|_1 < \epsilon + \|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j)\|_1$ with $\epsilon > 0$, we can calculate the gradient in Eq. 9 with:

$$\begin{aligned}\frac{\partial\overline{\mathcal{O}}(\mathbf{W})}{\partial\mathbf{W}} = G\Big(H_{is} - H_{ij}\Big)\cdot G\Big(-H_{is} + H_{ij}\Big)\cdot \\ (\frac{\partial H_{is}}{\partial\mathbf{W}} - \frac{\partial H_{ij}}{\partial\mathbf{W}}) + \lambda\Big(\frac{\partial T_{is}}{\mathbf{W}} + \frac{\partial T_{js}}{\mathbf{W}} - \frac{\partial T_{ij}}{\mathbf{W}}\Big) - \mu\mathbf{W}\end{aligned} \tag{10}$$

where $H_{ab} = \|\overline{\mathbf{h}}_a - \overline{\mathbf{h}}_b\|_1$ and $T_{ab} = \|\overline{\mathbf{h}}_a - \overline{\mathbf{h}}_b\|_2^2$, $\frac{\partial H_{ab}}{\partial\mathbf{W}}$ is given by:

$$\begin{aligned}\frac{\partial H_{ab}}{\partial\mathbf{W}} =&(\boldsymbol{x}_a - \boldsymbol{u})[\mathrm{sgn}(\overline{\mathbf{h}}_a - \overline{\mathbf{h}}_b)\odot(1 - \overline{\mathbf{h}}_a^2)]^\top - \\ &(\boldsymbol{x}_b - \boldsymbol{u})[\mathrm{sgn}(\overline{\mathbf{h}}_a - \overline{\mathbf{h}}_b)\odot(1 - \overline{\mathbf{h}}_b^2)]^\top,\end{aligned} \tag{11}$$

where $\odot$ represents Hadamard product (*i.e.*, element-wise product).

$\frac{\partial T_{ab}}{\partial\mathbf{W}}$ is given by

$$\begin{aligned}\frac{\partial T_{ab}}{\partial\mathbf{W}} =&(\boldsymbol{x}_a - \boldsymbol{u})[(\overline{\mathbf{h}}_a - \overline{\mathbf{h}}_b)\odot(1 - \overline{\mathbf{h}}_a^2)]^\top - \\ &(\boldsymbol{x}_b - \boldsymbol{u})[(\overline{\mathbf{h}}_a - \overline{\mathbf{h}}_b)\odot(1 - \overline{\mathbf{h}}_b^2)]^\top.\end{aligned} \tag{12}$$

Table 1: The detailed statistics of two datasets.

Datasets	SUN397	YouTube Faces
♯ Queries	1,800	6,500
♯ Database samples	106,953	614,626
♯ Classes	397	1,595
♯ Dimensions	1,600	1,770

With the gradient in Eq. 10, we can conduct stochastic gradient ascent as following:

$$\mathbf{W} = \mathbf{W} + \alpha\frac{\partial\overline{\mathcal{O}}(\mathbf{W})}{\partial\mathbf{W}} \tag{13}$$

where α is the learning rate. The detailed optimization procedure is provided in Algorithm 1. ϵ is set to be 1 in all our experiments.

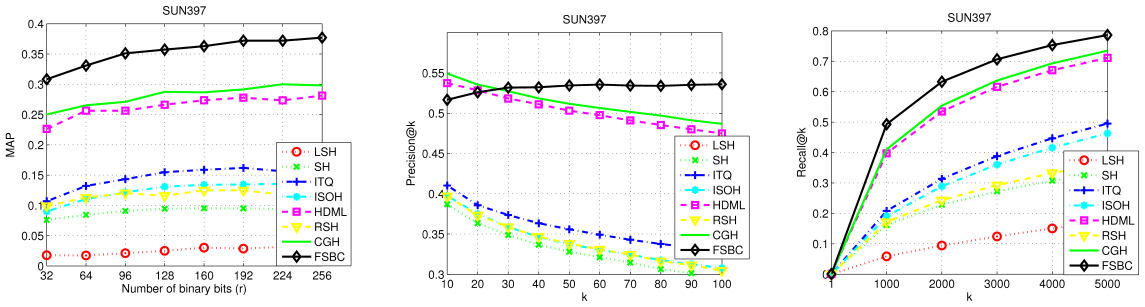
Note that our optimization procedure in Algorithm 1 is similar to standard stochastic gradient descent (SGD). The difference is that in each iteration, either than randomly selecting a subset of training examples, we only select the most extreme example from a subset of training examples for optimization. It is very plausible that this approach will work and the empirical results confirm our intuition that this should give good results. We are aware that the theoretical guarantees for standard SGD do not apply directly. It will be an interesting problem to provide theoretical guarantees for Algorithm 1.

4 Experiment

In this section, we first describe two datasets and the setting for our empirical study. Then we introduce the three evaluation metrics used in our experiments. Finally we compare the proposed Fast Structural Binary Coding (FSBC) against several state-of-the-art binary coding and hashing algorithms to demonstrate its effectiveness for large scale image search. Among these baseline approaches, four are unsupervised approaches, including one randomized method, Locality Sensitive Hashing (LSH) [Andoni and Indyk, 2008], one spectral approach, Spectral Hashing (SH) [Weiss *et al.*, 2008], and two linear projection techniques, Iterative Quantization (ITQ) [Gong *et al.*, 2012] and Isotropic Hashing (ISOH) [Kong and Li, 2012]. The other three are supervised approaches which use triplets to encode the label information (similar to our setting); they are Hamming Distance Metric Learning (HDML) [Norouzi *et al.*, 2012], Column Generation Hashing (CGH) [Li *et al.*, 2013], and Ranking-based Supervised Hashing (RSH) [Wang *et al.*, 2013].

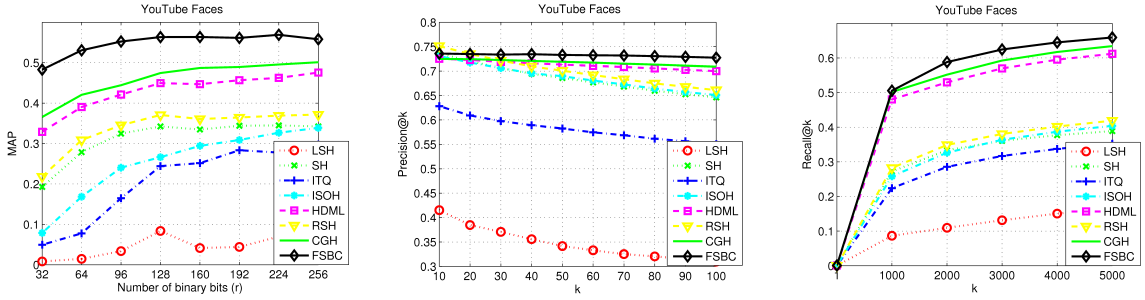
4.1 Datasets and setup

In the experiments, we perform image search over two different datasets, *i.e.*, SUN397 [Xiao *et al.*, 2010] and YouTube Faces [Wolf *et al.*, 2011]. SUN397 consists of about 108K images from 397 scene categories. In SUN397, each image is represented by a 1,600-dimensional feature vector extracted by principle component analysis (PCA) from 12,288-dimensional Deep Convolutional Activation Features [Gong *et al.*, 2014]. The YouTube Faces dataset contains 614,626 face images of 1,595 different people. In YouTube Faces,



(a) MAP vs. r on SUN397 (b) Precision@k on SUN397 (c) Recall@k on SUN397

Figure 3: (a) MAP vs. a varying number of binary bits ($r = \{32, 64, 96, 128, 160, 192, 224, 256\}$) for different binary coding and hashing algorithms on SUN397. (b) Precision@k on SUN397 when $r = 256$. (c) Recall@k on SUN397 when $r = 256$.



(a) MAP vs. r on YouTube Faces (b) Precision@k on YouTube Faces (c) Recall@k on YouTube Faces

Figure 4: (a) MAP vs. a varying number of binary bits ($r = \{32, 64, 96, 128, 160, 192, 224, 256\}$) for different binary coding and hashing algorithms on YouTube Faces. (b) Precision@k on YouTube Faces when $r = 256$. (c) Recall@k on YouTube Faces when $r = 256$.

each face image is represented by a 1,770-dimensional LBP feature vector [Ahonen *et al.*, 2006]. The detailed statistics of these two datasets are shown in Table 1.

In SUN397, 100 images are randomly sampled from each of the 18 largest scene categories to form a test set of 1,800 query images. For unsupervised approaches, all the database samples are used for training. For supervised methods, we randomly choose 200 images from each of the 18 scene categories to form a training set of 3,600 images; an additional 50 images from each of these 18 scene categories are randomly selected to form a validation set of 900 query images. All the rest of the images in the 397 categories are then used as the database samples. In YouTube Faces, 100 face images from each of the 65 largest face classes are randomly sampled to form a test set of 6,500 query images. For unsupervised learning, all the database images are used for training. For supervised learning, 1,000 images from each of the 65 face classes are randomly draw to form a training set of 65,000 face images; an additional 50 images from each of these 65 scene categories are randomly selected to form a validation set of 3250 query images. All the rest of the face images in the 1,595 face classes are treated as the database samples for retrieval.

We implement the proposed FSBC and baseline algorithms using Matlab on a PC with Intel Core i7-4770K Processor 3.5GHz and 32GB RAM. The parameters λ and μ of FSBC are determined by cross validation over the grid $\{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. We will discuss the parameter sensitivity later.

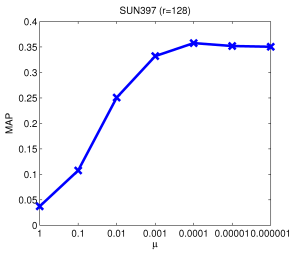
To measure the effectiveness of various binary coding and hashing techniques for image search, we consider three evaluation metrics, *i.e.*, Mean Average Precision (MAP), precision at top-k positions (Precision@k), and recall at top-k positions (Recall@k).

4.2 Results

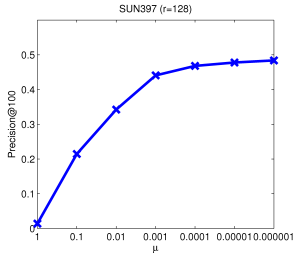
We compare the proposed Fast Structural Binary Coding (FSBC) against seven binary coding and hashing algorithms based upon SUN397 and YouTube Faces when r varies from 32 bits to 256 bits, as shown in Figure 3(a) and 4(a), respectively. We observe that with the increment of bits (r), FSBC consistently outperforms all baseline approaches for MAP. This is because FSBC not only optimizes the precision at the top of a Hamming distance ranking list, but also approximately preserves the geometric relationship between database examples in the original Euclidean space. For baseline approaches, supervised methods, *i.e.*, HDML, RSH, and CGH generally outperform unsupervised techniques since they can produce discriminative binary codes by incorporating label information appropriately. Among the unsupervised approaches, we notice that SH, ITQ and ISOH consistently outperform LSH. This suggests that utilizing underlying data structures, distributions, or topological information can produce more effective codes for image search tasks. The de-

Table 2: Image search performance (**MAP** and **Precision@100**) on **SUN397** and **YouTube Faces** when $r = 256$. All training times are recorded in second. The best **MAP** or **Precision@100** is displayed in bold-face type.

Algorithms	SUN397			YouTube Faces		
	MAP	**Prec@100**	**Training Time**	**MAP**	**Prec@100**	**Training Time**
LSH [Andoni and Indyk, 2008]	0.0310	0.1042	2.03	0.0845	0.3097	3.82×10^{2}
SH [Weiss *et al.*, 2008]	0.0968	0.2947	6.16×10^{1}	0.3422	0.6461	9.52×10^{2}
ITQ [Gong *et al.*, 2012]	0.1581	0.3289	5.62×10^{1}	0.2976	0.5504	6.60×10^{2}
ISOH [Kong and Li, 2012]	0.1385	0.3078	1.16×10^{1}	0.3387	0.6510	2.41×10^{2}
HDML [Norouzi *et al.*, 2012]	0.2814	0.4748	5.31×10^{3}	0.4755	0.6999	5.16×10^{3}
RSH [Wang *et al.*, 2013]	0.1376	0.3049	1.07×10^{3}	0.3604	0.6611	1.22×10^{3}
CGH [Li *et al.*, 2013]	0.2982	0.4868	4.34×10^{3}	0.5012	0.7090	5.95×10^{3}
FSBC	**0.3770**	**0.5359**	2.48×10^{3}	**0.5579**	**0.7273**	3.94×10^{3}

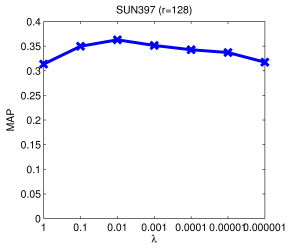


(a) MAP vs. μ

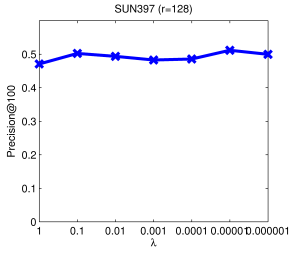


(b) Precision@100 vs. μ

Figure 5: Parameter sensitivity study of $\mu = \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ on SUN397 with $r = 128$ and $\lambda = 0.01$.



(a) MAP vs. λ



(b) Precision@100 vs. λ

Figure 6: Parameter sensitivity study of $\lambda = \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ on SUN397 with $r = 128$ and $\mu = 10^{-5}$.

tailed image search performances in terms of MAP and Precision@100 over the two datasets are provided in Table 2.

We also compare FSBC with baseline methods based upon Precision@k and Recall@k over SUN397 and YouTube Faces (when r is fixed as 256 bits). In Figure 3(b), 3(c), 4(b), and 4(c), we notice that FSBC generally outperforms all baseline approaches when k varies from 10 to 100 for Precision@k and as k varies from 0 to 5000 for Recall@k. This suggests that optimizing the objective of FSBC can significantly improve top-k image search performance. Note that Precision@k does not perform well when $k < 30$ in Figure 3(b) and 4(b). This may be because we only optimize the objective over a small random subset (p) of training examples in each iteration (for efficiency), not on the entire training set. We observed that as p increases, the performance ($k < 30$) will improve and may outperform baseline methods.

The training time of the proposed FSBC and baseline algorithms over the two datasets are provided in Tables 1. We observe that the offline training time of FSBC is less than those of HDML and CGH which all use label information in the form of triplet-level ranks. This is because FSBC is only optimized over the most extreme triplet while HDML and CGH weigh each triplet equally. For binary code generation, the main computational cost of FSBC depends on the linear projection and binarization operations. Hence, the test time of FSBC is 1.94×10^{-5} second for SUN397 and 2.26×10^{-5} second for YouTube Faces which is as efficient as typical linear binary coding or hashing algorithms.

We study the parameter sensitivity for FSBC (when $r = 128$) in Figure 5 and 6. In Figure 5, we observe that when λ is fixed at 0.01, the performance (MAP and Precision@100) of FSBC is relatively stable when μ varies from 10^{-3} to 10^{-6}. In Figure 6, we notice that when μ is fixed as 10^{-5}, the Precision@100 of FSBC is relatively robust when λ varies from 1 to 10^{-6} but MAP decreases as λ decreases from 10^{-2} to 10^{-6}. These results justify the effectiveness of the regularization term being used for preserving the geometric relationship between the database examples in the original Euclidean space.

5 Conclusion

In this paper, we proposed Fast Structural Binary Coding (FSBC) to explicitly optimize the precision at the top of a Hamming distance ranking list and approximately preserve the geometric relationship between the database examples in the original Euclidean space. The key idea is to train disciplined coding functions by optimizing a lower bound of AUC and penalize this objective such that similar database examples in the original Euclidean space can be returned as a whole in the Hamming distance ranking list. To find such a coding function, we relaxed the original discrete optimization objective with a continuous surrogate, and then derived a stochastic gradient descent method to optimize the surrogate objective. Empirical studies based upon two image datasets demonstrated that the proposed FSBC can outperform the state-of-the-arts for large scale image search.

References

[Ahonen *et al.*, 2006] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.

[Andoni and Indyk, 2008] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[Broder *et al.*, 1998] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permuations. In *Proceedings of ACM Symposium on Theory of Computing*, 1998.

[Gong *et al.*, 2012] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2012.

[Gong *et al.*, 2014] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proceedings of European Conference on Computer Vision*, 2014.

[Kong and Li, 2012] W. Kong and W.-J. Li. Istropic hashing. In *Proceedings of Advances in Neural Information Processing Systems 25*, 2012.

[Kulis and Darrell, 2009] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Proceedings of Advances in Neural Information Processing Systems 22*, 2009.

[Li *et al.*, 2013] X. Li, G. Lin, C. Shen, A. Hengel, and A. Dick. Learning hash functions using column generation. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[Liu *et al.*, 2012] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2012.

[Liu *et al.*, 2014] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *Proceedings of Advances in Neural Information Processing Systems 27*, 2014.

[Norouzi and Fleet, 2011] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[Norouzi *et al.*, 2012] M. Norouzi, D. J. Fleet, and R. Salakhutdinov. Hamming distance metric learning. In *Proceedings of Advances in Neural Information Processing Systems 25*, 2012.

[Song *et al.*, 2015a] D. Song, W. Liu, R. Ji, D. A. Meyer, and J. Smith. Top rank supervised binary coding for visual search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1922–1930, Santiago, Chile, 2015.

[Song *et al.*, 2015b] D. Song, W. Liu, D. A. Meyer, D. Tao, and R. Ji. Rank preserving hashing for rapid image search. In *Proceedings of Data Compression Conference*, pages 353–362, Snowbird, Utah, USA, 2015.

[Song *et al.*, 2015c] D. Song, D. A. Meyer, and D. Tao. Efficient latent link recommendation in signed networks. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1105–1114, Sydney, Australia, 2015.

[Wang *et al.*, 2013] J. Wang, W. Liu, A. X. Sun, and Y. Jiang. Learning hash codes with listwise supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[Wang *et al.*, 2014] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. In *arXiv:1408.2927*, 2014.

[Wang *et al.*, 2016] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data - a survey. *To appear in Proceedings of the IEEE*, 104(1):34–57, Jan 2016.

[Weiss *et al.*, 2008] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proceedings of Advances in Neural Information Processing Systems 21*, 2008.

[Weston and Blitzer, 2012] J. Weston and J. Blitzer. Latent structured ranking. In *Proceedings of Uncertainty in Artificial Interlligence*, 2012.

[Wolf *et al.*, 2011] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[Xiao *et al.*, 2010] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2010.

[Zhang *et al.*, 2014] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *Proceedings of the 30th International Conference on Machine Learning*, 2014.