Circulant Binary Embedding

Felix X. Yu[1] YUXINNAN@EE.COLUMBIA.EDU
Sanjiv Kumar[2] SANJIVK@GOOGLE.COM
Yunchao Gong[3] YUNCHAO@CS.UNC.EDU
Shih-Fu Chang[1] SFCHANG@EE.COLUMBIA.EDU
[1]Columbia University, New York, NY 10027
[2]Google Research, New York, NY 10011
[3]University of North Carolina at Chapel Hill, Chapel Hill, NC 27599

Abstract

Binary embedding of high-dimensional data requires long codes to preserve the discriminative power of the input space. Traditional binary coding methods often suffer from very high computation and storage costs in such a scenario. To address this problem, we propose Circulant Binary Embedding (CBE) which generates binary codes by projecting the data with a circulant matrix. The circulant structure enables the use of Fast Fourier Transformation to speed up the computation. Compared to methods that use unstructured matrices, the proposed method improves the time complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d \log d)$, and the space complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$ where d is the input dimensionality. We also propose a novel time-frequency alternating optimization to learn data-dependent circulant projections, which alternatively minimizes the objective in original and Fourier domains. We show by extensive experiments that the proposed approach gives much better performance than the state-of-the-art approaches for fixed time, and provides much faster computation with no performance degradation for fixed number of bits.

1. Introduction

Embedding input data in binary spaces is becoming popular for efficient retrieval and learning on massive data sets (Li et al., 2011; Gong et al., 2013a; Raginsky & Lazebnik, 2009; Gong et al., 2012; Liu et al., 2011). Moreover, in a large number of application domains such as computer vision, biology and finance, data is typically high-dimensional. When representing such high dimensional data by binary codes, it has been shown that long codes are required in order to achieve good performance. In fact, the required number of bits is $\mathcal{O}(d)$, where d is the input dimensionality (Li et al., 2011; Gong et al., 2013a; Sánchez & Perronnin, 2011). The goal of binary embedding is to well approximate the input distance as Hamming distance so that efficient learning and retrieval can happen directly in the binary space. It is important to note that another related area called *hashing* is a special case with slightly different goal: creating hash tables such that points that are similar fall in the same (or nearby) bucket with high probability. In fact, even in hashing, if high accuracy is desired, one typically needs to use hundreds of hash tables involving tens of thousands of bits.

Most of the existing linear binary coding approaches generate the binary code by applying a projection matrix, followed by a binarization step. Formally, given a data point, $\mathbf{x} \in \mathbb{R}^d$, the k-bit binary code, $h(\mathbf{x}) \in \{+1, -1\}^k$ is generated simply as

$$h(\mathbf{x}) = \text{sign}(\mathbf{R}\mathbf{x}), \tag{1}$$

where $\mathbf{R} \in \mathbb{R}^{k \times d}$, and $\text{sign}(\cdot)$ is a binary map which returns element-wise sign[1]. Several techniques have been proposed to generate the projection matrix randomly without taking into account the input data (Charikar, 2002; Raginsky & Lazebnik, 2009). These methods are very popular due to their simplicity but often fail to give the best performance due to their inability to adapt the codes with respect to the input data. Thus, a number of data-dependent techniques have been proposed with different optimization criteria such as reconstruction error (Kulis & Darrell, 2009), data dissimilarity (Norouzi & Fleet, 2012; Weiss et al.,

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

[1] A few methods transform the linear projection via a nonlinear map before taking the sign (Weiss et al., 2008; Raginsky & Lazebnik, 2009).

2008), ranking loss (Norouzi et al., 2012), quantization error after PCA (Gong et al., 2013b), and pairwise misclassification (Wang et al., 2010). These methods are shown to be effective for learning compact codes for relatively low-dimensional data. However, the $\mathcal{O}(d^2)$ computational and space costs prohibit them from being applied to learning long codes for high-dimensional data. For instance, to generate $\mathcal{O}(d)$-bit binary codes for data with $d \sim$1M, a huge projection matrix will be required needing TBs of memory, which is not practical[2].

In order to overcome these computational challenges, Gong et al. (2013a) proposed a bilinear projection based coding method for high-dimensional data. It reshapes the input vector $\mathbf{x}$ into a matrix $\mathbf{Z}$, and applies a bilinear projection to get the binary code:

$$h(\mathbf{x}) = \text{sign}(\mathbf{R}_1^T \mathbf{Z} \mathbf{R}_2). \tag{2}$$

When the shapes of $\mathbf{Z}, \mathbf{R}_1, \mathbf{R}_2$ are chosen appropriately, the method has time and space complexity of $\mathcal{O}(d^{1.5})$ and $\mathcal{O}(d)$, respectively. Bilinear codes make it feasible to work with datasets with very high dimensionality and have shown good results in a variety of tasks.

In this work, we propose a novel Circulant Binary Embedding (CBE) technique which is even faster than the bilinear coding. It is achieved by imposing a circulant structure on the projection matrix $\mathbf{R}$ in (1). This special structure allows us to use Fast Fourier Transformation (FFT) based techniques, which have been extensively used in signal processing. The proposed method further reduces the time complexity to $\mathcal{O}(d \log d)$, enabling efficient binary embedding for very high-dimensional data[3]. Table 1 compares the time and space complexity for different methods. This work makes the following contributions:

- We propose the circulant binary embedding method, which has space complexity $\mathcal{O}(d)$ and time complexity $\mathcal{O}(d \log d)$ (Section 2, 3).
- We propose to learn the data-dependent circulant projection matrix by a novel and efficient time-frequency alternating optimization, which alternatively optimizes the objective in the original and frequency domains (Section 4).
- Extensive experiments show that, compared to the state-of-the-art, the proposed method improves the result dramatically for a fixed time cost, and provides much faster computation with no performance degradation for a fixed number of bits (Section 5).

Method	Time	Space	Time (Learning)
Full projection	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(nd^3)$
Bilinear proj.	$\mathcal{O}(d^{1.5})$	$\mathcal{O}(d)$	$\mathcal{O}(nd^{1.5})$
Circulant proj.	$\mathcal{O}(d \log d)$	$\mathcal{O}(d)$	$\mathcal{O}(nd \log d)$

Table 1. Comparison of the proposed method (Circulant proj.) with other methods for generating long codes (code dimension k comparable to input dimension d). n is the number of instances used for learning data-dependent projection matrices.

2. Circulant Binary Embedding (CBE)

A circulant matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ is a matrix defined by a vector $\mathbf{r} = (r_0, r_2, \cdots, r_{d-1})^T$ (Gray, 2006)[4].

$$\mathbf{R} = \text{circ}(\mathbf{r}) := \begin{bmatrix} r_0 & r_{d-1} & \dots & r_2 & r_1 \\ r_1 & r_0 & r_{d-1} & & r_2 \\ \vdots & r_1 & r_0 & \ddots & \vdots \\ r_{d-2} & & \ddots & \ddots & r_{d-1} \\ r_{d-1} & r_{d-2} & \dots & r_1 & r_0 \end{bmatrix}. \tag{3}$$

Let $\mathbf{D}$ be a diagonal matrix with each diagonal entry being a Bernoulli variable (± 1 with probability 1/2). For $\mathbf{x} \in \mathbb{R}^d$, its d-bit Circulant Binary Embedding (CBE) with $\mathbf{r} \in \mathbb{R}^d$ is defined as:

$$h(\mathbf{x}) = \text{sign}(\mathbf{R}\mathbf{D}\mathbf{x}), \tag{4}$$

where $\mathbf{R} = \text{circ}(\mathbf{r})$. The k-bit ($k < d$) CBE is defined as the first k elements of $h(\mathbf{x})$. The need for such a $\mathbf{D}$ is discussed in Section 3. Note that applying $\mathbf{D}$ to $\mathbf{x}$ is equivalent to applying random sign flipping to each dimension of $\mathbf{x}$. Since sign flipping can be carried out as a preprocessing step for each input $\mathbf{x}$, here onwards for simplicity we will drop explicit mention of $\mathbf{D}$. Hence the binary code is given as $h(\mathbf{x}) = \text{sign}(\mathbf{R}\mathbf{x})$.

The main advantage of circulant binary embedding is its ability to use Fast Fourier Transformation (FFT) to speed up the computation.

Proposition 1. *For d-dimensional data, CBE has space complexity $\mathcal{O}(d)$, and time complexity $\mathcal{O}(d \log d)$.*

Since a circulant matrix is defined by a single column/row, clearly the storage needed is $\mathcal{O}(d)$. Given a data point $\mathbf{x}$, the d-bit CBE can be efficiently computed as follows. Denote $\circledast$ as operator of circulant convolution. Based on the definition of circulant matrix,

$$\mathbf{R}\mathbf{x} = \mathbf{r} \circledast \mathbf{x}. \tag{5}$$

The above can be computed based on Discrete Fourier Transformation (DFT), for which fast algorithm (FFT) is available. The DFT of a vector $\mathbf{t} \in \mathbb{C}^d$ is a d-dimensional vector with each element defined as

[2] In principle, one can generate the random entries of the matrix on-the-fly (with fixed seeds) without needing to store the matrix. But this will increase the computational time even further.

[3] One could in principal use other structured matrices like Hadamard matrix along with a sparse random Gaussian matrix to achieve fast projection as was done in fast Johnson-Lindenstrauss transform(Ailon & Chazelle, 2006; Dasgupta et al., 2011), but it is still slower than circulant projection and needs more space.

[4] The circulant matrix is sometimes equivalently defined by "circulating" the rows instead of the columns.

$$\mathcal{F}(\mathbf{t})_l = \sum_{m=0}^{d-1} t_n \cdot e^{-i2\pi lm/d}, l = 0, \cdots, d-1. \quad (6)$$

The above can be expressed equivalently in a matrix form as

$$\mathcal{F}(\mathbf{t}) = \mathbf{F}_d \mathbf{t}, \quad (7)$$

where $\mathbf{F}_d$ is the d-dimensional DFT matrix. Let $\mathbf{F}_d^H$ be the conjugate transpose of $\mathbf{F}_d$. It is easy to show that $\mathbf{F}_d^{-1} = (1/d)\mathbf{F}_d^H$. Similarly, for any $\mathbf{t} \in \mathbb{C}^d$, the Inverse Discrete Fourier Transformation (IDFT) is defined as

$$\mathcal{F}^{-1}(\mathbf{t}) = (1/d)\mathbf{F}_d^H \mathbf{t}. \quad (8)$$

Since the convolution of two signals in their original domain is equivalent to the hadamard product in their frequency domain (Oppenheim et al., 1999),

$$\mathcal{F}(\mathbf{R}\mathbf{x}) = \mathcal{F}(\mathbf{r}) \circ \mathcal{F}(\mathbf{x}). \quad (9)$$

Therefore,

$$h(\mathbf{x}) = \text{sign}\left(\mathcal{F}^{-1}(\mathcal{F}(\mathbf{r}) \circ \mathcal{F}(\mathbf{x}))\right). \quad (10)$$

For k-bit CBE, $k < d$, we only need to pick the first k bits of $h(\mathbf{x})$. As DFT and IDFT can be efficiently computed in $\mathcal{O}(d \log d)$ with FFT (Oppenheim et al., 1999), generating CBE has time complexity $\mathcal{O}(d \log d)$.

3. Randomized Circulant Binary Embedding

A simple way to obtain CBE is by generating the elements of $\mathbf{r}$ in (3) independently from the standard normal distribution $\mathcal{N}(0, 1)$. We call this method randomized CBE (CBE-rand). A desirable property of any embedding method is its ability to approximate input distances in the embedded space. Suppose $\mathcal{H}_k(\mathbf{x}_1, \mathbf{x}_2)$ is the normalized Hamming distance between k-bit codes of a pair of points $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$:

$$\mathcal{H}_k(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{k}\sum_{i=0}^{k-1}\left|\text{sign}(\mathbf{R}_{i\cdot}\mathbf{x}_1) - \text{sign}(\mathbf{R}_{i\cdot}\mathbf{x}_2)\right|/2, \quad (11)$$

and $\mathbf{R}_{i\cdot}$ is the i-th row of $\mathbf{R}$, $\mathbf{R} = \text{circ}(\mathbf{r})$. If $\mathbf{r}$ is sampled from $\mathcal{N}(0, 1)$, from (Charikar, 2002),

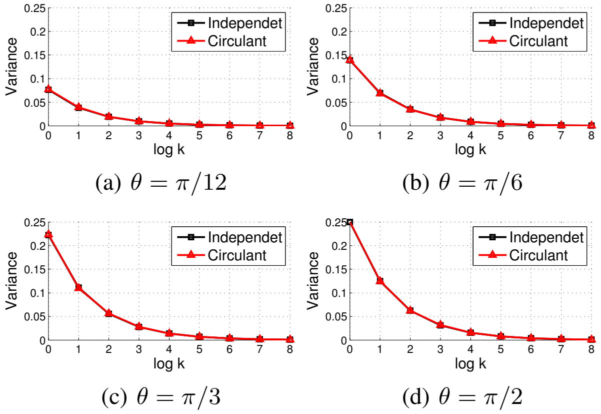
$$\mathbf{Pr}\left(\text{sign}(\mathbf{r}^T\mathbf{x}_1) \neq \text{sign}(\mathbf{r}^T\mathbf{x}_2)\right) = \theta/\pi, \quad (12)$$

where θ is the angle between $\mathbf{x}_1$ and $\mathbf{x}_2$. Since all the vectors that are circulant variants of $\mathbf{r}$ also follow the same distribution, it is easy to see that

$$\mathbf{E}(\mathcal{H}_k(\mathbf{x}_1, \mathbf{x}_2)) = \theta/\pi. \quad (13)$$

For the sake of discussion, if k projections, *i.e.*, first k rows of $\mathbf{R}$, were generated independently, it is easy to show that the variance of $\mathcal{H}_k(\mathbf{x}_1, \mathbf{x}_2)$ will be

$$\mathbf{Var}(\mathcal{H}_k(\mathbf{x}_1, \mathbf{x}_2)) = \theta(\pi - \theta)/k\pi^2. \quad (14)$$



(a) $\theta = \pi/12$ (b) $\theta = \pi/6$

(c) $\theta = \pi/3$ (d) $\theta = \pi/2$

Figure 1. The *analytical* variance of normalized hamming distance of independent bits as in (14), and the *sample* variance of normalized hamming distance of circulant bits, as a function of angle between points (θ) and number of bits (k). The two curves overlap.

Thus, with more bits (larger k), the normalized hamming distance will be close to the expected value, with lower variance. In other words, the normalized hamming distance approximately preserves the angle[5]. Unfortunately in CBE, the projections are the rows of $\mathbf{R} = \text{circ}(\mathbf{r})$, which are not independent. This makes it hard to derive the variance analytically. To better understand CBE-rand, we run simulations to compare the analytical variance of normalized hamming distance of independent projections (14), and the *sample* variance of normalized hamming distance of circulant projections in Figure 1. For each θ and k, we randomly generate $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ such that their angle is θ[6]. We then generate k-dimensional code with CBE-rand, and compute the hamming distance. The variance is estimated by applying CBE-rand 1,000 times. We repeat the whole process 1,000 times, and compute the averaged variance. Surprisingly, the curves of "Independent" and "Circulant" variances are almost indistinguishable. This means that bits generated by CBE-rand are generally as good as the independent bits for angle preservation. An intuitive explanation is that for the circulant matrix, though all the rows are dependent, circulant shifting one or multiple elements will in fact result in very different projections in most cases. We will later show in experiments on real-world data that CBE-rand and Locality Sensitive Hashing (LSH)[7] has almost identical performance (yet CBE-rand is significantly faster) (Section 5).

[5] In this paper, we consider the case that the data points are ℓ_2 normalized. Therefore the cosine distance, *i.e.*, 1 - $\cos(\theta)$, is equivalent to the l_2 distance.

[6] This can be achieved by extending the 2D points $(1, 0)$, $(\cos\theta, \sin\theta)$ to d-dimension, and performing a random orthonormal rotation, which can be formed by the Gram-Schmidt process on random vectors.

[7] Here, by LSH we imply the binary embedding using $\mathbf{R}$ such that all the rows of $\mathbf{R}$ are sampled iid. With slight abuse of notation, we still call it "hashing" following (Charikar, 2002).

Note that the distortion in input distances after circulant binary embedding comes from two sources: circulant projection, and binarization. For the circulant projection step, recent works have shown that the Johnson-Lindenstrauss-type lemma holds with a slightly worse bound on the number of projections needed to preserve the input distances with high probability (Hinrichs & Vybíral, 2011; Zhang & Cheng, 2013; Vybíral, 2011; Krahmer & Ward, 2011). These works also show that before applying the circulant projection, an additional step of randomly flipping the signs of input dimensions is necessary[8]. To show why such a step is required, let us consider the special case when $\mathbf{x}$ is an all-one vector, $\mathbf{1}$. The circulant projection with $\mathbf{R} = \text{circ}(\mathbf{r})$ will result in a vector with all elements to be $\mathbf{r}^T\mathbf{1}$. When $\mathbf{r}$ is independently drawn from $\mathcal{N}(0, 1)$, this will be close to 0, and the norm cannot be preserved. Unfortunately the Johnson-Lindenstrauss-type results do not generalize to the distortion caused by the binarization step.

One problem with the randomized CBE method is that it does not utilize the underlying data distribution while generating the matrix $\mathbf{R}$. In the next section, we propose to learn $\mathbf{R}$ in a data-dependent fashion, to minimize the distortions due to circulant projection and binarization.

4. Learning Circulant Binary Embedding

We propose data-dependent CBE (CBE-opt), by optimizing the projection matrix with a novel time-frequency alternating optimization. We consider the following objective function in learning the d-bit CBE. The extension of learning $k < d$ bits will be shown in Section 4.2.

$$\begin{aligned} \underset{\mathbf{B},\mathbf{r}}{\text{argmin}} \quad & ||\mathbf{B} - \mathbf{X}\mathbf{R}^T||_F^2 + \lambda||\mathbf{R}\mathbf{R}^T - \mathbf{I}||_F^2 \\ \text{s.t.} \quad & \mathbf{R} = \text{circ}(\mathbf{r}), \end{aligned} \tag{15}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$, is the data matrix containing n training points: $\mathbf{X} = [\mathbf{x}_0, \cdots, \mathbf{x}_{n-1}]^T$, and $\mathbf{B} \in \{-1, 1\}^{n\times d}$ is the corresponding binary code matrix.[9]

In the above optimization, the first term minimizes distortion due to binarization. The second term tries to make the projections (rows of $\mathbf{R}$, and hence the corresponding bits) as uncorrelated as possible. In other words, this helps to reduce the redundancy in the learned code. If $\mathbf{R}$ were to be an orthogonal matrix, the second term will vanish and the optimization would find the best rotation such that the distortion due to binarization is minimized. However, when $\mathbf{R}$ is a circulant matrix, $\mathbf{R}$, in general, will not be orthogonal. Similar objective has been used in previous works including (Gong et al., 2013b;a) and (Wang et al., 2010).

4.1. The Time-Frequency Alternating Optimization

The above is a combinatorial optimization problem, for which an optimal solution is hard to find. In this section we propose a novel approach to efficiently find a local solution. The idea is to alternatively optimize the objective by fixing $\mathbf{r}$, and $\mathbf{B}$, respectively. For a fixed $\mathbf{r}$, optimizing $\mathbf{B}$ can be easily performed in the input domain ("time" as opposed to "frequency"). For a fixed $\mathbf{B}$, the circulant structure of $\mathbf{R}$ makes it difficult to optimize the objective in the input domain. Hence we propose a novel method, by optimizing $\mathbf{r}$ in the frequency domain based on DFT. This leads to a very efficient procedure.

For a fixed $\mathbf{r}$. The objective is independent on each element of $\mathbf{B}$. Denote B_{ij} as the element of the i-th row and j-th column of $\mathbf{B}$. It is easy to show that $\mathbf{B}$ can be updated as:

$$\begin{aligned} B_{ij} = & \begin{cases} 1 & \text{if } \mathbf{R}_{j\cdot}\mathbf{x}_i \geq 0 \\ -1 & \text{if } \mathbf{R}_{j\cdot}\mathbf{x}_i < 0 \end{cases}, \\ & i = 0, \cdots, n-1. \quad j = 0, \cdots, d-1. \end{aligned} \tag{16}$$

For a fixed $\mathbf{B}$. Define $\tilde{\mathbf{r}}$ as the DFT of the circulant vector $\tilde{\mathbf{r}} := \mathcal{F}(\mathbf{r})$. Instead of solving $\mathbf{r}$ directly, we propose to solve $\tilde{\mathbf{r}}$, from which $\mathbf{r}$ can be recovered by IDFT.

Key to our derivation is the fact that DFT projects the signal to a set of orthogonal basis. Therefore the ℓ_2 norm can be preserved. Formally, according to Parseval's theorem , for any $\mathbf{t} \in \mathbb{C}^d$ (Oppenheim et al., 1999),

$$||\mathbf{t}||_2^2 = (1/d)||\mathcal{F}(\mathbf{t})||_2^2.$$

Denote diag$(\cdot)$ as the diagonal matrix formed by a vector. Denote $\Re(\cdot)$ and $\Im(\cdot)$ as the real and imaginary parts, respectively. We use $\mathbf{B}_{i\cdot}$ to denote the i-th row of $\mathbf{B}$. With complex arithmetic, the first term in (15) can be expressed in the frequency domain as:

$$||\mathbf{B} - \mathbf{X}\mathbf{R}^T||_F^2 = \frac{1}{d}\sum_{i=0}^{n-1}||\mathcal{F}(\mathbf{B}_{i\cdot}^T - \mathbf{R}\mathbf{x}_i)||_2^2 \tag{17}$$

$$= \frac{1}{d}\sum_{i=0}^{n-1}||\mathcal{F}(\mathbf{B}_{i\cdot}^T) - \tilde{\mathbf{r}} \circ \mathcal{F}(\mathbf{x}_i)||_2^2 = \frac{1}{d}\sum_{i=0}^{n-1}||\mathcal{F}(\mathbf{B}_{i\cdot}^T) - \text{diag}(\mathcal{F}(\mathbf{x}_i))\tilde{\mathbf{r}}||_2^2$$

$$= \frac{1}{d}\sum_{i=0}^{n-1}\Big(\mathcal{F}(\mathbf{B}_{i\cdot}^T) - \text{diag}(\mathcal{F}(\mathbf{x}_i))\tilde{\mathbf{r}}\Big)^T\Big(\mathcal{F}(\mathbf{B}_{i\cdot}^T) - \text{diag}(\mathcal{F}(\mathbf{x}_i))\tilde{\mathbf{r}}\Big)$$

$$= \frac{1}{d}\Big[\Re(\tilde{\mathbf{r}})^T\mathbf{M}\Re(\tilde{\mathbf{r}}) + \Im(\tilde{\mathbf{r}})^T\mathbf{M}\Im(\tilde{\mathbf{r}}) + \Re(\tilde{\mathbf{r}})^T\mathbf{h} + \Im(\tilde{\mathbf{r}})^T\mathbf{g}\Big] + ||\mathbf{B}||_F^2,$$

where,

$$\mathbf{M} = \text{diag}\Big(\sum_{i=0}^{n-1}\Re(\mathcal{F}(\mathbf{x}_i)) \circ \Re(\mathcal{F}(\mathbf{x}_i)) + \Im(\mathcal{F}(\mathbf{x}_i)) \circ \Im(\mathcal{F}(\mathbf{x}_i))\Big)$$

$$\mathbf{h} = -2\sum_{i=0}^{n-1}\Re(\mathcal{F}(\mathbf{x}_i)) \circ \Re(\mathcal{F}(\mathbf{B}_{i\cdot}^T)) + \Im(\mathcal{F}(\mathbf{x}_i)) \circ \Im(\mathcal{F}(\mathbf{B}_{i\cdot}^T))$$

$$\mathbf{g} = 2\sum_{i=0}^{n-1}\Im(\mathcal{F}(\mathbf{x}_i)) \circ \Re(\mathcal{F}(\mathbf{B}_{i\cdot}^T)) - \Re(\mathcal{F}(\mathbf{x}_i)) \circ \Im(\mathcal{F}(\mathbf{B}_{i\cdot}^T)).$$

[8] For each dimension, whether the sign needs to be flipped is predetermined by a ($p = 0.5$) Bernoulli variable.

[9] If the data is ℓ_2 normalized, we can set $\mathbf{B} \in \{-1/\sqrt{d}, 1/\sqrt{d}\}^{n\times d}$ to make $\mathbf{B}$ and $\mathbf{X}\mathbf{R}^T$ more comparable. This does not empirically influence the performance.

For the second term in (15), we note that the circulant matrix can be diagonalized by DFT matrix $\mathbf{F}_d$ and its conjugate transpose $\mathbf{F}_d^H$. Formally, for $\mathbf{R} = \mathrm{circ}(\mathbf{r})$, $\mathbf{r} \in \mathbb{R}^d$,

$$\mathbf{R} = (1/d)\mathbf{F}_d^H \mathrm{diag}(\mathcal{F}(\mathbf{r}))\mathbf{F}_d. \tag{18}$$

Let $\mathrm{Tr}(\cdot)$ be the trace of a matrix. Therefore,

$$\begin{aligned}
&||\mathbf{R}\mathbf{R}^T - \mathbf{I}||_F^2 = ||\frac{1}{d}\mathbf{F}_d^H(\mathrm{diag}(\tilde{\mathbf{r}})^H\mathrm{diag}(\tilde{\mathbf{r}}) - \mathbf{I})\mathbf{F}_d||_F^2\\
&= \mathrm{Tr}\Big[\frac{1}{d}\mathbf{F}_d^H(\mathrm{diag}(\tilde{\mathbf{r}})^H\mathrm{diag}(\tilde{\mathbf{r}}) – \mathbf{I})^H(\mathrm{diag}(\tilde{\mathbf{r}})^H\mathrm{diag}(\tilde{\mathbf{r}}) – \mathbf{I})\mathbf{F}_d\Big]\\
&= \mathrm{Tr}\left[(\mathrm{diag}(\tilde{\mathbf{r}})^H\mathrm{diag}(\tilde{\mathbf{r}}) - \mathbf{I})^H(\mathrm{diag}(\tilde{\mathbf{r}})^H\mathrm{diag}(\tilde{\mathbf{r}}) - \mathbf{I})\right]\\
&= ||\tilde{\mathbf{r}}^H \circ \tilde{\mathbf{r}} - \mathbf{1}||_2^2 = ||\Re(\tilde{\mathbf{r}})^2 + \Im(\tilde{\mathbf{r}})^2 - \mathbf{1}||_2^2.
\end{aligned} \tag{19}$$

Furthermore, as $\mathbf{r}$ is real-valued, additional constraints on $\tilde{\mathbf{r}}$ are needed. For any $u \in \mathbb{C}$, denote $\overline{u}$ as the complex conjugate of u. We have the following result (Oppenheim et al., 1999): For any real-valued vector $\mathbf{t} \in \mathbb{C}^d$, $\mathcal{F}(\mathbf{t})_0$ is real-valued, and

$$\mathcal{F}(\mathbf{t})_{d-i} = \overline{\mathcal{F}(\mathbf{t})_i}, \quad i = 1, \cdots, \lfloor d/2 \rfloor.$$

From (17) $-$ (19), the problem of optimizing $\tilde{\mathbf{r}}$ becomes

$$\begin{aligned}
\underset{\tilde{\mathbf{r}}}{\mathrm{argmin}} \quad & \Re(\tilde{\mathbf{r}})^T\mathbf{M}\Re(\tilde{\mathbf{r}}) + \Im(\tilde{\mathbf{r}})^T\mathbf{M}\Im(\tilde{\mathbf{r}}) + \Re(\tilde{\mathbf{r}})^T\mathbf{h}\\
& + \Im(\tilde{\mathbf{r}})^T\mathbf{g} + \lambda d||\Re(\tilde{\mathbf{r}})^2 + \Im(\tilde{\mathbf{r}})^2 - \mathbf{1}||_2^2 \qquad (20)\\
\text{s.t.} \quad & \Im(\tilde{r}_0) = 0\\
& \Re(\tilde{r}_i) = \Re(\tilde{r}_{d-i}), i = 1, \cdots, \lfloor d/2 \rfloor\\
& \Im(\tilde{r}_i) = -\Im(\tilde{r}_{d-i}), i = 1, \cdots, \lfloor d/2 \rfloor.
\end{aligned}$$

The above is non-convex. Fortunately, the objective function can be decomposed, such that we can solve two variables at a time. Denote the diagonal vector of the diagonal matrix $\mathbf{M}$ as $\mathbf{m}$. The above optimization can then be decomposed to the following sets of optimizations.

$$\underset{\tilde{r}_0}{\mathrm{argmin}} \ m_0\tilde{r}_0^2 + h_0\tilde{r}_0 + \lambda d\left(\tilde{r}_0^2 - 1\right)^2, \text{ s.t. } \tilde{r}_0 = \overline{\tilde{r}_0}. \tag{21}$$

$$\begin{aligned}
\underset{\tilde{r}_i}{\mathrm{argmin}} \ & (m_i + m_{d-i})(\Re(\tilde{r}_i)^2 + \Im(\tilde{r}_i)^2) \qquad (22)\\
& + 2\lambda d\left(\Re(\tilde{r}_i)^2 + \Im(\tilde{r}_i)^2 - 1\right)^2\\
& + (h_i + h_{d-i})\Re(\tilde{r}_i) + (g_i - g_{d-i})\Im(\tilde{r}_i),\\
& i = 1, \cdots, \lfloor d/2 \rfloor.
\end{aligned}$$

In (21), we need to minimize a 4^{th} order polynomial with one variable, with the closed form solution readily available. In (22), we need to minimize a 4^{th} order polynomial with two variables. Though the closed form solution is hard (requiring solution of a cubic bivariate system), we can find local minima by gradient descent, which can be considered as having constant running time for such small-scale problems. The overall objective is guaranteed to be non-increasing in each step. In practice, we can get a good solution with just 5-10 iterations. In summary, the proposed time-frequency alternating optimization procedure has running time $\mathcal{O}(nd \log d)$.

4.2. Learning $k < d$ Bits

In the case of learning $k < d$ bits, we need to solve the following optimization problem:

$$\begin{aligned}
\underset{\mathbf{B},\mathbf{r}}{\mathrm{argmin}} \quad & ||\mathbf{B}\mathbf{P}_k - \mathbf{X}\mathbf{P}_k^T\mathbf{R}^T||_F^2 + \lambda||\mathbf{R}\mathbf{P}_k\mathbf{P}_k^T\mathbf{R}^T - \mathbf{I}||_F^2\\
\text{s.t.} \quad & \mathbf{R} = \mathrm{circ}(\mathbf{r}),
\end{aligned} \tag{23}$$

in which $\mathbf{P}_k = \begin{bmatrix} \mathbf{I}_k & \mathbf{O} \\ \mathbf{O} & \mathbf{O}_{d-k} \end{bmatrix}$, $\mathbf{I}_k$ is a $k \times k$ identity matrix, and $\mathbf{O}_{d-k}$ is a $(d-k) \times (d-k)$ all-zero matrix.

In fact, the right multiplication of $\mathbf{P}_k$ can be understood as a "temporal cut-off", which is equivalent to a frequency domain convolution. This makes the optimization difficult, as the objective in frequency domain can no longer be decomposed. To address this issues, we propose a simple solution in which $B_{ij} = 0, i = 0, \cdots, n-1, j = k, \cdots, d-1$ in (15). Thus, the optimization procedure remains the same, and the cost is also $\mathcal{O}(nd \log d)$. We will show in experiments that this heuristic provides good performance in practice.

5. Experiments

To compare the performance of the proposed circulant binary embedding technique, we conducted experiments on three real-world high-dimensional datasets used by the current state-of-the-art method for generating long binary codes (Gong et al., 2013a). The Flickr-25600 dataset contains 100K images sampled from a noisy Internet image collection. Each image is represented by a $25,600$ dimensional vector. The ImageNet-51200 contains 100k images sampled from 100 random classes from ImageNet (Deng et al., 2009), each represented by a $51,200$ dimensional vector. The third dataset (ImageNet-25600) is another random subset of ImageNet containing 100K images in $25,600$ dimensional space. All the vectors are normalized to be of unit norm.

We compared the performance of the randomized (CBE-rand) and learned (CBE-opt) versions of our circulant embeddings with the current state-of-the-art for high-dimensional data, *i.e.*, bilinear embeddings. We use both the randomized (bilinear-rand) and learned (bilinear-opt) versions. Bilinear embeddings have been shown to perform similar or better than another promising technique called Product Quantization (Jegou et al., 2011). Finally, we also compare against the binary codes produced by the baseline LSH method (Charikar, 2002), which is still applicable to 25,600 and 51,200 dimensional feature but with much longer running time and much more space. We also show an experiment with relatively low-dimensional data in 2048 dimensional space using Flickr data to compare against techniques that perform well for low-dimensional data but do not scale to high-dimensional scenario. Exam-

d	Full proj.	Bilinear proj.	Circulant proj.
2^{15}	5.44×10^2	2.85	1.11
2^{17}	-	1.91×10^1	4.23
2^{20} (1M)	-	3.76×10^2	3.77×10^1
2^{24}	-	1.22×10^4	8.10×10^2
2^{27} (100M)	-	2.68×10^5	8.15×10^3

Table 2. Computational time (ms) of full projection (LSH, ITQ, SH *etc*.), bilinear projection (Bilinear), and circulant projection (CBE). The time is based on a single 2.9GHz CPU core. The error is within 10%. An empty cell indicates that the memory needed for that method is larger than the machine limit of 24GB.

ple techniques include ITQ (Gong et al., 2013b), SH (Weiss et al., 2008), SKLSH (Raginsky & Lazebnik, 2009), and AQBC (Gong et al., 2012).

Following (Gong et al., 2013a; Norouzi & Fleet, 2012; Gordo & Perronnin, 2011), we use 10,000 randomly sampled instances for training. We then randomly sample 500 instances, different from the training set as queries. The performance (recall@1-100) is evaluated by averaging the recalls of the query instances. The ground-truth of each query instance is defined as its 10 nearest neighbors based on ℓ_2 distance. For each dataset, we conduct two sets of experiments: *fixed-time* where code generation time is fixed and *fixed-bits* where the number of bits is fixed across all techniques. We also show an experiment where the binary codes are used for classification.

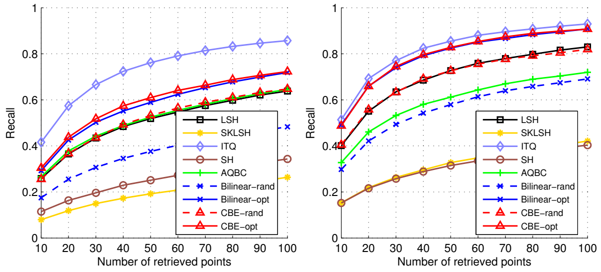
The proposed CBE method is found robust to the choice of λ in (15). For example, in the retrieval experiments, the performance difference for λ = 0.1, 1, 10, is within 0.5%. Therefore, in all the experiments, we simply fix λ = 1. For the bilinear method, in order to get fast computation, the feature vector is reshaped to a near-square matrix, and the dimension of the two bilinear projection matrices are also chosen to be close to square. Parameters for other techniques are tuned to give the best results on these datasets.

Computational Time. When generating k-bit code for d-dimensional data, the full projection, bilinear projection, and circulant projection methods have time complexity $O(kd)$, $O(\sqrt{k}d)$, and $O(d \log d)$, respectively. We compare the computational time in Table 2 on a fixed hardware. Based on our implementation, the computational time of the above three methods can be roughly characterized as $d^2 : d\sqrt{d} : 5d \log d$. Note that faster implementation of FFT algorithms will lead to better computational time for CBE by further reducing the constant factor. Due to the small storage requirement $\mathcal{O}(d)$, and the wide availability of highly optimized FFT libraries, CBE is also suitable for implementation on GPU. Our preliminary tests based on GPU shows up to 20 times speedup compared to CPU. In this paper, for fair comparison, we use same CPU based implementation for all the methods.

Retrieval. The recall for different methods is compared on the three datasets in Figure 2, 3, and 4 respectively. The top row in each figure shows the performance for different methods when the code generation time for all the methods is kept the same as that of CBE. For a fixed time, the proposed CBE yields much better recall than other methods. Even CBE-rand outperforms LSH and Bilinear code by a large margin. The second row compares the performance for different techniques with codes of same length. In this case, the performance of CBE-rand is almost identical to LSH even though it is hundreds of time faster. This is consistent with our analysis in Section 3. Moreover, CBE-opt/CBE-rand outperform the Bilinear-opt/Bilinear-rand in addition to being 2-3 times faster.

Original	LSH	Bilinear-opt	CBE-opt
25.59±0.33	23.49±0.24	24.02±0.35	24.55 ±0.30

Table 3. Multiclass classification accuracy on binary coded ImageNet-25600. The binary codes of same dimensionality are 32 times more space efficient than the original features (single-float).

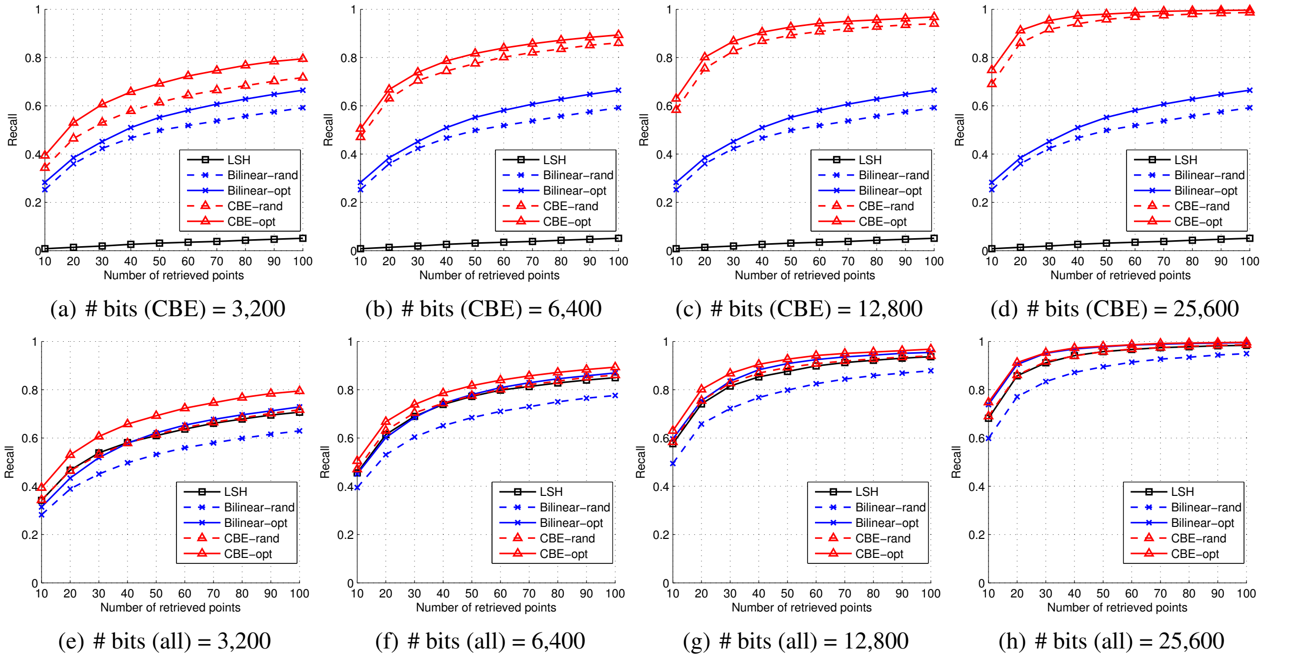


(a) # bits = 1,024 (b) # bits = 2,048

Figure 5. Performance comparison on relatively low-dimensional data (Flickr-2048) with fixed number of bits. CBE gives comparable performance to the state-of-the-art even on low-dimensional data as the number of bits is increased. However, note that these other methods do not scale to very high-dimensional data setting which is the main focus of this work.

Classification. Besides retrieval, we also test the binary codes for classification. The advantage is to save on storage allowing even large scale datasets to fit in memory (Li et al., 2011; Sánchez & Perronnin, 2011). We follow the asymmetric setting of (Sánchez & Perronnin, 2011) by training linear SVM on binary code sign($\mathbf{R}\mathbf{x}$), and testing on the original $\mathbf{R}\mathbf{x}$. This empirically has been shown to give better accuracy than the symmetric procedure. We use ImageNet-25600, with randomly sampled 100 images per category for training, 50 for validation and 50 for testing. The code dimension is set as 25,600. As shown in Table 3, CBE, which has much faster computation, does not show any performance degradation compared to LSH or bilinear codes in classification task as well.

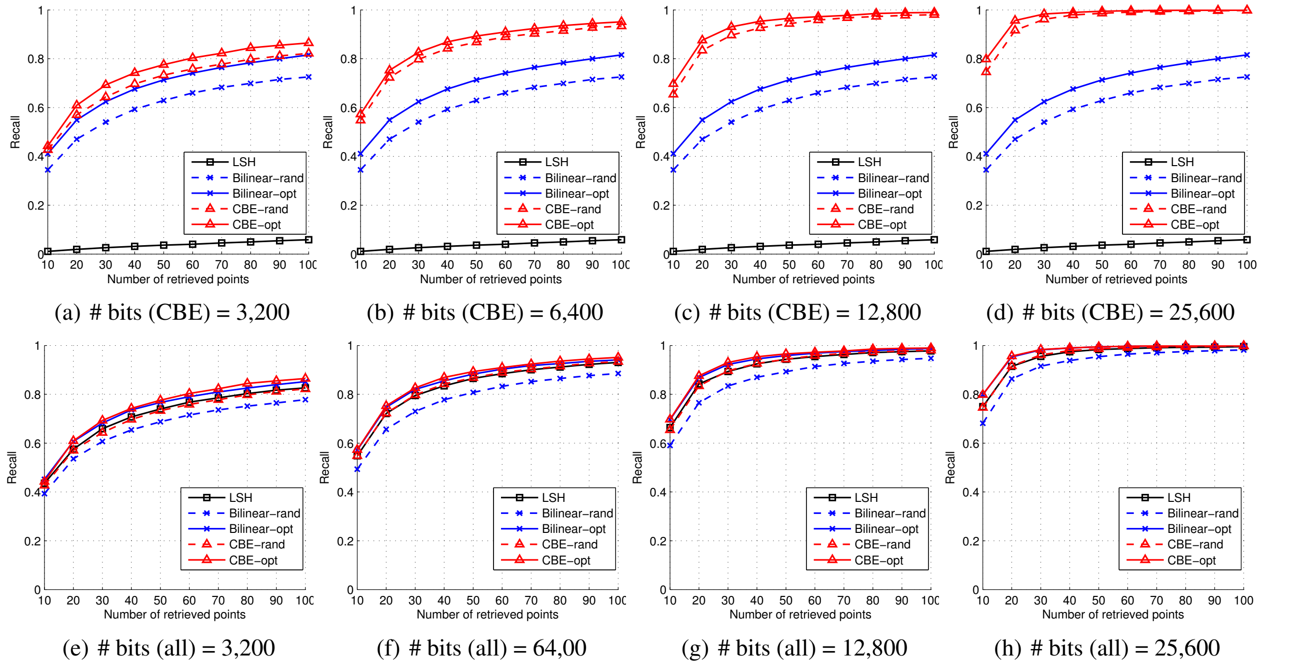
Low-Dimensional Experiment. There exist several techniques that do not scale to high-dimensional case. To compare our method with those, we conducted experiments



(a) # bits (CBE) = 3,200 (b) # bits (CBE) = 6,400 (c) # bits (CBE) = 12,800 (d) # bits (CBE) = 25,600

(e) # bits (all) = 3,200 (f) # bits (all) = 6,400 (g) # bits (all) = 12,800 (h) # bits (all) = 25,600

Figure 2. Recall on Flickr-25600. The standard deviation is within 1%. **First Row**: Fixed time. "# bits" is the number of bits of CBE. Other methods are using less bits to make their computational time identical to CBE. **Second Row**: Fixed number of bits. CBE-opt/CBE-rand are 2-3 times faster than Bilinear-opt/Bilinear-rand, and hundreds of times faster than LSH.

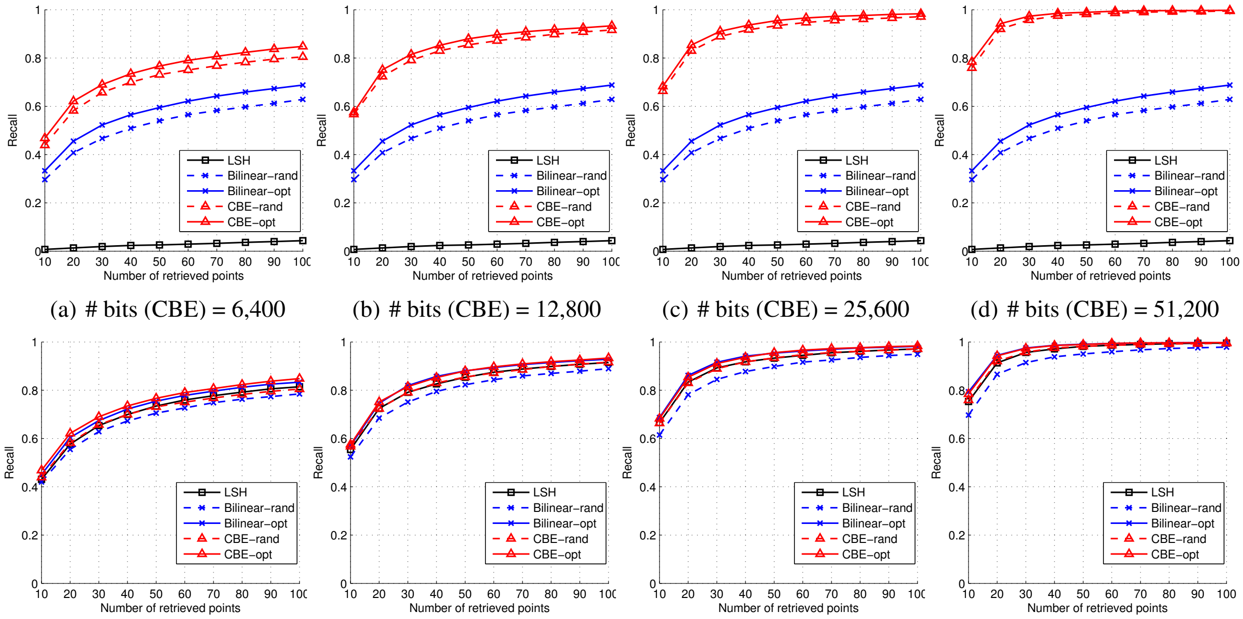


(a) # bits (CBE) = 3,200 (b) # bits (CBE) = 6,400 (c) # bits (CBE) = 12,800 (d) # bits (CBE) = 25,600

(e) # bits (all) = 3,200 (f) # bits (all) = 64,00 (g) # bits (all) = 12,800 (h) # bits (all) = 25,600

Figure 3. Recall on ImageNet-25600. The standard deviation is within 1%. **First Row**: Fixed time. "# bits" is the number of bits of CBE. Other methods are using less bits to make their computational time identical to CBE. **Second Row**: Fixed number of bits. CBE-opt/CBE-rand are 2-3 times faster than Bilinear-opt/Bilinear-rand, and hundreds of times faster than LSH.

with fixed number of bits on a relatively low-dimensional dataset (Flickr-2048), constructed by randomly sampling 2,048 dimensions of Flickr-25600. As shown in Figure 5, though CBE is not designed for such scenario, the CBE-opt performs better or equivalent to other techniques except ITQ which scales very poorly with d ($\mathcal{O}(d^3)$). Moreover, as the number of bits increases, the gap between ITQ and CBE becomes much smaller suggesting that the performance of ITQ is not expected to be better than CBE even if one could run ITQ on high-dimensional data.



(a) # bits (CBE) = 6,400 (b) # bits (CBE) = 12,800 (c) # bits (CBE) = 25,600 (d) # bits (CBE) = 51,200

(e) # bits (all) = 6,400 (f) # bits (all) = 12,800 (g) # bits (all) = 25,600 (h) # bits (all) = 51,200

Figure 4. Recall on ImageNet-51200. The standard deviation is within 1%. **First Row**: Fixed time. "# bits" is the number of bits of CBE. Other methods are using less bits to make their computational time identical to CBE. **Second Row**: Fixed number of bits. CBE-opt/CBE-rand are 2-3 times faster than Bilinear-opt/Bilinear-rand, and hundreds of times faster than LSH.

6. Semi-supervised Extension

In some applications, one can have access to a few labeled pairs of similar and dissimilar data points. Here we show how the CBE formulation can be extended to incorporate such information in learning. This is achieved by adding an additional objective term $J(\mathbf{R})$.

$$\begin{aligned} &\operatorname*{argmin}_{\mathbf{B},\mathbf{r}} ||\mathbf{B}-\mathbf{X}\mathbf{R}^T||_F^2+\lambda||\mathbf{R}\mathbf{R}^T-\mathbf{I}||_F^2+\mu J(\mathbf{R}) \qquad (24)\\ &\text{s.t.} \quad \mathbf{R}=\operatorname{circ}(\mathbf{r}), \end{aligned}$$

$$J(\mathbf{R})=\sum_{i,j\in\mathcal{M}}||\mathbf{R}\mathbf{x}_i-\mathbf{R}\mathbf{x}_j||_2^2-\sum_{i,j\in\mathcal{D}}||\mathbf{R}\mathbf{x}_i-\mathbf{R}\mathbf{x}_j||_2^2. \qquad (25)$$

Here $\mathcal{M}$ and $\mathcal{D}$ are the set of "similar" and "dissimilar" instances, respectively. The intuition is to maximize the distances between the dissimilar pairs, and minimize the distances between the similar pairs. Such a term is commonly used in semi-supervised binary coding methods (Wang et al., 2010). We again use the time-frequency alternating optimization procedure of Section 4. For a fixed $\mathbf{r}$, the optimization procedure to update $\mathbf{B}$ is the same. For a fixed $\mathbf{B}$, optimizing $\mathbf{r}$ is done in frequency domain by expanding $J(\mathbf{R})$ as below, with similar techniques used in Section 4.

$$||\mathbf{R}\mathbf{x}_i-\mathbf{R}\mathbf{x}_j||_2^2=(1/d)||\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j))\tilde{\mathbf{r}}||_2^2.$$

Therefore,

$$J(\mathbf{R})=(1/d)(\Re(\tilde{\mathbf{r}})^T\mathbf{A}\Re(\tilde{\mathbf{r}})+\Im(\tilde{\mathbf{r}})^T\mathbf{A}\Im(\tilde{\mathbf{r}})), \qquad (26)$$

where, $\mathbf{A}=\mathbf{A}_1+\mathbf{A}_2-\mathbf{A}_3-\mathbf{A}_4$, and

$$\begin{aligned} \mathbf{A}_1&=\sum_{(i,j)\in\mathcal{M}}\Re(\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j)))^T\Re(\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j))),\\ \mathbf{A}_2&=\sum_{(i,j)\in\mathcal{M}}\Im(\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j)))^T\Im(\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j))),\\ \mathbf{A}_3&=\sum_{(i,j)\in\mathcal{D}}\Re(\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j)))^T\Re(\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j))),\\ \mathbf{A}_4&=\sum_{(i,j)\in\mathcal{D}}\Im(\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j)))^T\Im(\operatorname{diag}(\mathcal{F}(\mathbf{x}_i)-\mathcal{F}(\mathbf{x}_j))). \end{aligned}$$

Hence, the optimization can be carried out as in Section 4, where $\mathbf{M}$ in (17) is simply replaced by $\mathbf{M}+\mu\mathbf{A}$. Our experiments show that the semi-supervised extension improves over the non-semi-supervised version by 2% in terms of averaged AUC on the ImageNet-25600 dataset.

7. Conclusion

We have proposed circulant binary embedding for generating long codes for very high-dimensional data. A novel time-frequency alternating optimization was also introduced to learn the model parameters from the training data. The proposed method has time complexity $\mathcal{O}(d\log d)$ and space complexity $\mathcal{O}(d)$, while showing no performance degradation on real-world data compared to more expensive approaches ($\mathcal{O}(d^2)$ or $\mathcal{O}(d^{1.5})$). On the contrary, for the fixed time, it showed significant accuracy gains. The full potential of the method can be unleashed when applied to ultra-high dimensional data (say d ∼100M), for which no other methods are applicable.

References

Ailon, Nir and Chazelle, Bernard. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *ACM Symposium on Theory of Computing*, 2006.

Charikar, Moses S. Similarity estimation techniques from rounding algorithms. In *ACM Symposium on Theory of Computing*, 2002.

Dasgupta, Anirban, Kumar, Ravi, and Sarlós, Tamás. Fast locality-sensitive hashing. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2011.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009.

Gong, Yunchao, Kumar, Sanjiv, Verma, Vishal, and Lazebnik, Svetlana. Angular quantization-based binary codes for fast similarity search. In *Advances in Neural Information Processing Systems*, 2012.

Gong, Yunchao, Kumar, Sanjiv, Rowley, Henry A, and Lazebnik, Svetlana. Learning binary codes for high-dimensional data using bilinear projections. In *Computer Vision and Pattern Recognition*, 2013a.

Gong, Yunchao, Lazebnik, Svetlana, Gordo, Albert, and Perronnin, Florent. 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, 2013b.

Gordo, Albert and Perronnin, Florent. Asymmetric distances for binary embeddings. In *Computer Vision and Pattern Recognition*, 2011.

Gray, Robert M. *Toeplitz and circulant matrices: A review*. Now Pub, 2006.

Hinrichs, Aicke and Vybíral, Jan. Johnson-Lindenstrauss lemma for circulant matrices. *Random Structures & Algorithms*, 39(3):391–398, 2011.

Jegou, Herve, Douze, Matthijs, and Schmid, Cordelia. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

Krahmer, Felix and Ward, Rachel. New and improved Johnson-Lindenstrauss embeddings via the restricted isometry property. *SIAM Journal on Mathematical Analysis*, 43(3):1269–1281, 2011.

Kulis, Brian and Darrell, Trevor. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems*, 2009.

Li, Ping, Shrivastava, Anshumali, Moore, Joshua, and Konig, Arnd Christian. Hashing algorithms for large-scale learning. In *Advances in Neural Information Processing Systems*, 2011.

Liu, Wei, Wang, Jun, Kumar, Sanjiv, and Chang, Shih-Fu. Hashing with graphs. In *International Conference on Machine Learning*, 2011.

Norouzi, Mohammad and Fleet, David. Minimal loss hashing for compact binary codes. *International Conference on Machine Learning*, 2012.

Norouzi, Mohammad, Fleet, David, and Salakhutdinov, Ruslan. Hamming distance metric learning. In *Advances in Neural Information Processing Systems*, 2012.

Oppenheim, Alan V, Schafer, Ronald W, Buck, John R, et al. *Discrete-time signal processing*, volume 5. Prentice Hall Upper Saddle River, 1999.

Raginsky, Maxim and Lazebnik, Svetlana. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems*, 2009.

Sánchez, Jorge and Perronnin, Florent. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition*, 2011.

Vybíral, Jan. A variant of the Johnson–Lindenstrauss lemma for circulant matrices. *Journal of Functional Analysis*, 260(4):1096–1105, 2011.

Wang, Jun, Kumar, Sanjiv, and Chang, Shih-Fu. Sequential projection learning for hashing with compact codes. In *International Conference on Machine Learning*, 2010.

Weiss, Yair, Torralba, Antonio, and Fergus, Rob. Spectral hashing. In *Advances in Neural Information Processing Systems*, 2008.

Zhang, Hui and Cheng, Lizhi. New bounds for circulant Johnson-Lindenstrauss embeddings. *arXiv preprint arXiv:1308.6339*, 2013.