Circulant Binary Embedding (CBE)

Felix X. Yu Columbia University
Sanjiv Kumar Google Research
Yunchao Gong Facebook AI Research
Shih-Fu Chang Columbia University

ICML, Beijing, June 24, 2014

Binary Embedding

- Transform the input data into binary code
- Given $\mathbf{x} \in \mathbb{R}^d$, $h(\mathbf{x}) \in \{1, -1\}^k$

Why binary embedding?

- Learning and retrieval can happen in the binary space
- Save storage and running time
- Widely used to speedup retrieval and classification [LSMK11, RL09]

Method

$$h(\mathbf{x}) = \text{sign}(\mathbf{R}\mathbf{x}), \quad \mathbf{R} \in \mathbb{R}^{k \times d}$$

- Randomized **R**: LSH [Cha02]
- Optimized **R**: reconstruction error [KD09], quantization error[GLGP13], pairwise similarity [WKC10] etc.

Binary Embedding (cont'd)

Difficulties for High-dimensional data

- High-dimensional data requires long code to accurately preserve the discriminative power [LSMK11] [GKRL13] [SP11]

$$k \sim \Theta(d)$$

- Computing the **full projection Rx**, $\mathbf{R} \in \mathbb{R}^{\Theta(d) \times d}$, has computational complexity and space complexity $\mathcal{O}(d^2)$
- $d \sim$ **1 Million: TBs of memory!**

How to efficiently perform binary embedding for high-dimensional data?

Binary Embedding (cont'd)

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

Related Work: Bilinear projection [GKRL13]

- Reshape $\mathbf{x} \in \mathbb{R}^d$ into a matrix $\mathbf{Z} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$
- Apply a bilinear projection to get the binary code

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

Our Approach: Circulant Binary Embedding (CBE)

- Much better retrieval performance for fixed coding time by allowing generating more bits
- Much faster computation with no performance degradation for fixed number of bits

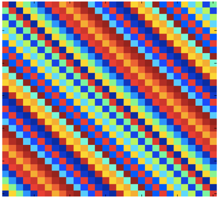
Table of Content

Circulant Binary Embedding (CBE)

$$h(\mathbf{x}) = \mathrm{sign}(\mathbf{RDx}), \quad \mathbf{R} \in \mathbb{R}^{d \times d}$$

- $\mathbf{R}$ is a **circulant matrix**
- $\mathbf{R}$ is defined by a vector $\mathbf{r} = (r_0, r_1, \cdots, r_{d-1})^T$

$$\mathbf{R} = \mathrm{circ}(\mathbf{r}) := \begin{bmatrix} r_0 & r_{d-1} & \cdots & r_2 & r_1 \\ r_1 & r_0 & r_{d-1} & & r_2 \\ \vdots & r_1 & r_0 & \ddots & \vdots \\ r_{d-2} & & \ddots & \ddots & r_{d-1} \\ r_{d-1} & r_{d-2} & \cdots & r_1 & r_0 \end{bmatrix}$$



- $\mathbf{D}$ is a diagonal matrix, each entry ± 1 with probability $1/2$ (random sign flipping, dropped to simplify notation)
- k-bit ($k < d$) code: first k elements of $h(\mathbf{x})$

Circulant Binary Embedding (CBE)

$$h(\mathbf{x}) = \text{sign}(\mathbf{R}\mathbf{x}), \quad \mathbf{R} = \text{circ}(\mathbf{r})$$

Why CBE?

FFT-based Computation

$$h(\mathbf{x}) = \text{sign}(\mathbf{R}\mathbf{x}), \quad \mathbf{R} = \text{circ}(\mathbf{r})$$

1. Circulant projection is identical to circular convolution

$$\mathbf{R}\mathbf{x} = \text{circ}(\mathbf{r})\mathbf{x} = \mathbf{r} \circledast \mathbf{x}$$

2. Circular convolution can be computed with FFT

$$\mathbf{r} \circledast \mathbf{x} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{r}) \circ \mathcal{F}(\mathbf{x}))$$

3. Time complexity $\mathcal{O}(d \log d)$. Space complexity $\mathcal{O}(d)$

Related: Johnson-Lindenstruss results with circulant and other structured matrices [AC06] [Vyb11]

How to choose R?

- Randomized CBE (CBE-rand)
- Learning data-dependent CBE (CBE-opt)

Randomized CBE

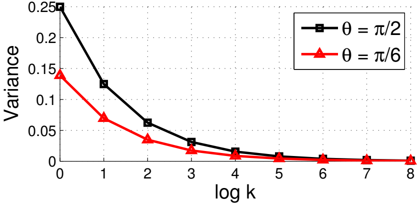
$$h(\mathbf{x}) = \text{sign}(\mathbf{Rx}), \quad \mathbf{R} = \text{circ}(\mathbf{r})$$

Each element of $\mathbf{r}$ is generated *i.i.d.* from $\mathcal{N}(0,1)$

Distance Perserving Properties

For any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, let θ be the angle between $\mathbf{x}_1, \mathbf{x}_2$

- $\mathbb{P}\left(h_i(\mathbf{x}_1) \neq h_i(\mathbf{x}_2)\right) = \theta/\pi$
- $\mathbb{E}\left(\frac{1}{k}\textbf{hamming}(h_{1\ldots k}(\mathbf{x}_1), h_{1\ldots k}(\mathbf{x}_2))\right) = \theta/\pi$
- $\textbf{Var}\left(\frac{1}{k}\textbf{hamming}(h_{1\ldots k}(\mathbf{x}_1), h_{1\ldots k}(\mathbf{x}_2))\right) = \quad ?$



Learning Data-dependent CBE

- Consider learning d-bits code first
- $\mathbf{X} \in \mathbb{R}^{n \times d}$: $\mathbf{X} = [\mathbf{x}_0, \cdots, \mathbf{x}_{n-1}]^T$
- $\mathbf{B} \in \{-1, 1\}^{n \times d}$: the binary code matrix

$$\underset{\mathbf{B}, \mathbf{r}}{\operatorname{argmin}} \quad \underbrace{||\mathbf{B} - \mathbf{X}\mathbf{R}^T||_F^2}_{\text{Binary distortion}} + \lambda \underbrace{||\mathbf{R}\mathbf{R}^T - \mathbf{I}||_F^2}_{\text{Non-redundancy in the bits}}$$

$$\text{s.t.} \quad \mathbf{R} = \operatorname{circ}(\mathbf{r})$$

- A challenging combinatorial optimization problem

Time-Frequency Alternating Minimization

Optimize $\mathbf{B}$ with fixed $\mathbf{r}$, in original "time" domain

$$\underset{\mathbf{B}}{\text{argmin}} \;\; ||\mathbf{B} - \mathbf{X}\mathbf{R}^T||_F^2, \quad \mathbf{B} = \text{sign}(\mathbf{X}\mathbf{R}^T)$$

Optimize $\mathbf{r}$ with fixed $\mathbf{B}$

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

- Can be solved efficiently in the frequency domain

Time-Frequency Alternating Minimization (cont'd)

- We optimize $\tilde{\mathbf{r}} := \mathcal{F}(\mathbf{r})$
- Key tool: Parseval's theorem (DFT preserves distance)

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

- Non-convex. Can be decomposed into d independent small optimization problems (4^{th} order polynomials with only 2 variables!)

Time-Frequency Alternating Minimization (cont'd)

Remarks on the algorithm

- The objective guaranteed to be non-increasing
- Good solution with just 5-10 iterations
- Running time $\mathcal{O}(nd \log d)$
- $\mathcal{O}(d)$ storage and parallel nature, suitable for GPU
- Not sensitive to λ

Learning $k < d$ bits

- A simple approach: setting the last $(d - k)$ bits to zero

Computational Time

Computational time based on a fixed hardware

d	Full projection	Bilinear projection	CBE
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^{27} (100M)	-	2.68×10^5	8.15×10^3

- Dramatic speedup for high-dim data
- Moderate speedup for low-dim data (FFT overhead)

Large-Scale Nearest-Neighbor Search

Methods

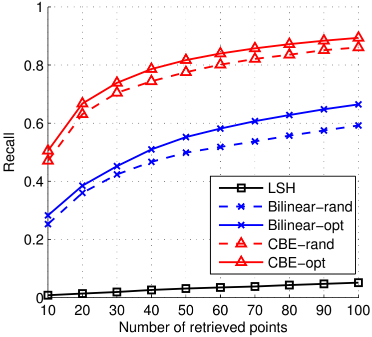
- CBE (CBE-rand, CBE-opt)
- LSH
- Bilinear code (Bilinear-rand, Bilinear-opt)

Experimental Setting

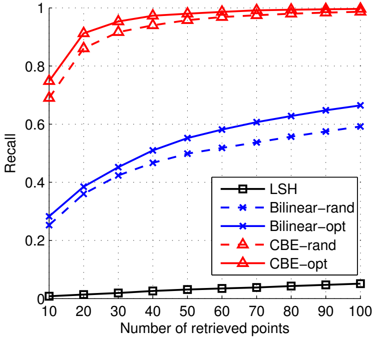
- 100k images, 25,600 dimensional feature
- Use an image as query to retrieve NN. Repeat 500 times
- Ground-truth: 10 nearest neighbors based on ℓ_2 distance

Large-Scale Nearest-Neighbor Search (cont'd)

Recall (fixed coding time):
Much higher recall than LSH and bilinear code



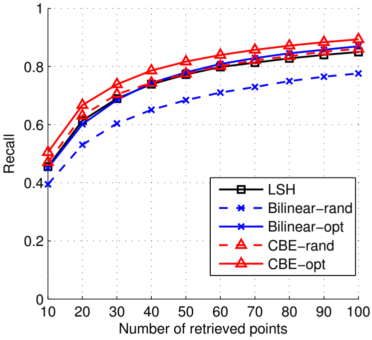
(a) # bits (CBE) = 6,400



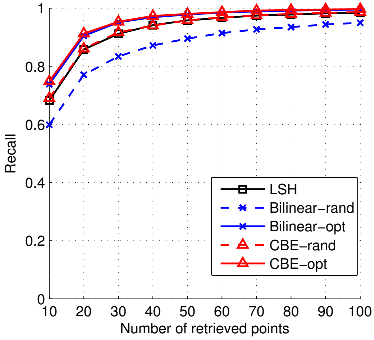
(b) # bits (CBE) = 25,600

Large-Scale Nearest-Neighbor Search (cont'd)

Recall (fixed number of bits):
Comparable or even better performance with faster computation



(c) # bits (all) = 6,400



(d) # bits (all) = 25,600

Large-Scale Classification

Learning on binary code:

- Advantage: save storage
- ImageNet data: 1k categories, 100 images per category for training, 50 for validation and 50 for testing
- $d = k = 25,600$ (32 times more space efficient)

Multiclass classification accuracy (%)

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

- CBE: faster computation, no performance degradation

Conclusion and More

Conclusion

- An $\mathcal{O}(d \log d)$ method for high-dimensional binary embedding
- Much better retrieval performance for fixed coding time
- Much faster computation for fixed number of bits
- CBE can be applied to data with $\sim$100M dimensions!

More

- CBE can be easily extended to **semi-supervised case**
- Implementation of CBE and baselines available at `https://github.com/felixyu/cbe`

The Requirement of $\mathbf{D}$

$$h(\mathbf{x}) = \mathrm{sign}(\mathbf{Rx}), \quad \mathbf{R} = \mathrm{circ}(\mathbf{r})$$

Two Types of Distance Distortions

1. Distortion from the circulant projection
   - Johnson-Lindenstruss type results with structured matrices [Vyb11]
   - The random sign flipping is required
   - If $\mathbf{x}$ is an all-1 vector, all the bits will be the same, and close to 0
2. Distortion from $\mathrm{sign}(\cdot)$

References I

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

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

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

Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin.
Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.

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

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

References II

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

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

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

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