Streaming Binary Sketching based on Subspace Tracking and Diagonal Uniformization

Anne Morvan ∗1,2, Antoine Souloumiac1, Cédric Gouy-Pailleur1, and Jamal Atif2

1CEA, LIST, 91191 Gif-sur-Yvette, France
2Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, 75016 Paris, France

May 23, 2017

Abstract

In this paper, we address the problem of learning compact similarity-preserving embeddings for massive high-dimensional streams of data in order to perform efficient similarity search. We present a new method for computing binary compressed representations - *sketches* - of high-dimensional real feature vectors. Given an expected code length c and high-dimensional input data points, our algorithm provides a binary code of c bits aiming at preserving the distance between the points from the original high-dimensional space. Our offline version of the algorithm outperforms the offline state-of-the-art methods regarding their computation time complexity and have a similar quality of the sketches. It also provides convergence guarantees. Moreover, our algorithm can be straightforwardly used in the streaming context by not requiring neither the storage of the whole dataset nor a chunk. We demonstrate the quality of our binary sketches through extensive experiments on real data for the nearest neighbors search task in the offline and online settings.

1 Introduction

In large-scale machine learning application fields such as computer vision, or metagenomics, learning similarity-preserving binary codes is critical in order to perform efficient indexing of large-scale high-dimensional data. Storage requirements can be reduced and similarity search sped up by embedding high-dimensional data into a compact binary code.

A classical method is Locality-Sensitive Hashing (LSH) (Andoni and Indyk [2008]) for nearest neighbors search: 1) input data with high dimension d are projected onto a lower c-dimensional space through a c × d random projection with i.i.d. Gaussian entries, 2) then a hashing function is applied to the resulting projected vector to obtain the final binary codes. Two examples for the hashing function are cross-polytope LSH (Terasawa and Tanaka [2007]) which returns the closest vector from the set {±1e_i}_{1 ≤ i ≤ c} where {e_i}_{1 ≤ i ≤ c} stands for the canonical basis, and hyperplane LSH (Andoni et al. [2015]) corresponding to the sign function applied pointwise. For reducing the storage cost of the projection and the computation time of the matrix-vector products (O(c × d)), a pseudo-random matrix with structure can be used instead (Andoni et al. [2015], Bojarski et al. [2017]) leading to a reduced time complexity of O(d log c) thanks to fast Hadamard and Fourier transforms.

In order to increase the accuracy of the data sketches in the context of nearest neighbors search or classification, this projection can be also learned from data (Weiss et al. [2008], Wang et al. [2012], Gong [2017])

∗To whom correspondence should be addressed: anne.morvan@cea.fr. Partly supported by the Direction Générale de l’Armement (French Ministry of Defense).
As Principal Component Analysis (PCA) is a common tool for reducing data dimensionality, PCA is often performed: data are then projected onto the first $c$ principal components. But the PCA alone is not sufficient. Indeed, the $c$ first principal components are chosen with a decreasing order of explained variance. It means that principal directions with higher variance carry more information. Consequently, associating each of the $c$ directions to one of the $c$ bits is equivalent to giving more weights to less informative directions and will lead to poor performance of the obtained sketches.

To remedy this problem, after data have been projected on the first principal components of the covariance matrix, a solution consists in applying a suitable rotation on the projected data before performing the hashing function, in order to balance variance over the principal components. In work from Jegou et al. [2010], a random rotation is successfully applied giving quite good results. In ITIterative Quantization (ITQ) (Gong and Lazebnik [2011], Gong et al. [2013]) or in Isotropic Hashing (IsoHash) (Kong and Li [2012]), the rotation is rather learned. In ITQ, the rotation is iteratively computed by an alternating minimization algorithm corresponding to an orthogonal Procrustes problem. This technique is currently the state-of-the-art for computing similarity-preserving binary codes.

ITQ method comes with some drawbacks, though. First, the convergence guarantees are only empirical. The method needs a parameter for the number of iterations required to compute the rotation. Second, the alternative optimization process relies on the computation of the SVD of a $c \times c$ matrix at each iteration, which even if $c$ is small in comparison with $d$, corresponds to a time complexity of $O(c^3)$. Finally, ITQ is a completely offline process since 1) the whole dataset needs to be stored for computing the $c$ principal components, 2) the computation of the rotation is based on the whole $c \times n$ projected dataset onto the principal components where $n$ is the number of instances. This can be prohibitive when dealing with lots of high-dimensional data.

**Contributions** In this paper, our contributions are two-fold: 1) We provide an offline algorithm outperforming ITQ from the following points of view: accuracy, computation time and spatial complexities, and convergence guarantees. 2) We also introduce a streaming adaptation of the algorithm where data is seen only once and principal subspace plus the balancing rotation are updated as new data is seen. Besides the $c \times d$ principal subspace to return, this requires only the storage of two $c \times c$ matrices - one is the covariance matrix of the projected data onto the $c$ principal components - instead of the whole initial and projected datasets as it is the case for ITQ. Depending on the availability of data and the expected accuracy of the sketches, the binary compact codes can be computed from one, two or three passes over the data. In any case, for only one pass over the data, our online algorithm gives better results than OSH (Leng et al. [2015a]) while being far less computationally demanding.

2 Related work

Two paradigms exist for constructing hash functions (Wang et al. [2016]): data-independent (Andoni and Indyk [2008], Raginsky and Lazebnik [2009], Grauman and Kulis [2011]) and data dependent methods. The latter ones learn the hash codes from a training set and perform better. The learning can be unsupervised (Weiss et al. [2008], Liu et al. [2011], Gong and Lazebnik [2011], Gong et al. [2013], Kong and Li [2012], Lee [2012], Liu et al. [2014], Yu et al. [2014], Raziperchikolaei and Carreira-Perpiñán [2016]) aiming at preserving distances in the original space or (semi-)supervised which also tries to preserve label similarity (Wang et al. [2012], Liu et al. [2012]). Some recent hashing functions involve deep learning (Lai et al. [2015], Chen et al. [2015]). When the dataset is too large to be loaded into memory, distributed (Leng et al. [2015b]) and online hashing techniques (Huang et al. [2013], Leng et al. [2015a]) have been developed. Online Hashing (OKH) (Huang et al. [2013]) learns the hash functions from a stream of pair of data with a "Passive-Aggressive" method. In Online Sketching Hashing (OSH) (Leng et al. [2015a]), the binary embeddings are learned from a maintained sketch of the dataset with a smaller size but preserving the property of interest. The proposed algorithm belongs to this latter category of online unsupervised hyperplanes-based hashing methods.
3 The proposed offline unsupervised model for binary quantization

3.1 Common unsupervised offline binary quantization problem statement

Let us first introduce some notations. We have a stream of \( n \) data points \( \{x_t \in \mathbb{R}^d\}_{1 \leq t \leq n} \) which constitute the columns of the data matrix \( X \in \mathbb{R}^{d \times n} \) supposed to be zero-centered. Our goal is to learn a binary code matrix \( B \in \{-1, 1\}^{c \times n} \) where \( c \) denotes the code length such that for each bit \( k = 1...c \), the binary encoding function is defined by \( h_k(x_t) = \text{sign}(\tilde{w}_k^T x_t) \) where \( \tilde{w}_k \) are column vectors of hyperplane coefficients and \( \text{sign}(x) = 1 \) if \( x \geq 0 \) and \( -1 \) otherwise which is applied component-wise on coefficients of vectors. \( B = \text{sign}(W X) \) where each row \( k \) of \( W \) is \( \tilde{w}_k^T \), with \( W \in \mathbb{R}^{c \times d} \).

In ITQ and our model, \( \tilde{W} = RW \) where \( R \) is a suitable \( c \times c \) orthogonal matrix and the \( c \times d \) matrix \( W \) represents the linear dimensionality reduction method applied to data. For instance, \( W \) can be the matrix whose row vectors \( w_k^T \) correspond to the \( c \) first principal components of the covariance matrix \( C = XX^T \) (an other supervised approach is to perform Canonical Correlation Analysis (Gong et al. [2013])). So the challenge is rather in defining an appropriate orthogonal matrix.

For ITQ, \( R \) is obtained through the optimization process minimizing the following quantization loss:

\[
Q(B, R) = ||B - \tilde{W} X||_F^2 = ||B - RW X||_F^2 = ||B - RV||_F^2
\]

where \( V = WX \) and \( ||.||_F \) denotes the Froebenius norm.

After having initialized \( R \) with a random orthogonal matrix thanks to a QR decomposition with time complexity \( O(c^3) \), ITQ alternatively computes \( B \) after freezing \( R \) and optimizes \( R \) according to \( B \) by performing the SVD of \( V^T B \). Algorithm 2 in appendix shows the procedure for obtaining a locally optimal rotation \( R \) after \( K \) iterations.

3.2 UnifDiag: the proposed diagonal uniformization-based offline method for learning a suitable rotation

Let the \( c \times c \) symmetric matrix \( \text{Cov} = V V^T \) be the covariance matrix of projected data \( V = WX \). In our offline model, \( R \) is learned to balance the variance over the \( c \) directions given by the \( c \) principal components of \( \text{Cov} \). Let us consider the \( c \) diagonal coefficients of \( \text{Cov} \): \( \sigma_1^2, ..., \sigma_c^2 \) s.t. \( \sigma_1^2 \geq \sigma_2^2 \geq ... \geq \sigma_c^2 \) \(^2\)

As \( \text{Cov} \) is symmetric, \( \text{Tr}(\text{Cov}) = \sum_{i=1}^{c} \sigma_i^2 = \sum_{i=1}^{c} \lambda_i \) where \( \text{Tr} \) stands for the Trace application and \( \lambda_1, ..., \lambda_c \) are the \( c \) first eigenvalues of \( C \) s.t. \( \lambda_1 \geq ... \geq \lambda_c \). Balancing variance over the \( c \) directions can be seen as equalizing the diagonal coefficients of \( \text{Cov} \) such that \( \sigma_1^2 = ... = \sigma_c^2 \stackrel{\text{def}}{=} \tau \). Figure 1 shows the covariance matrices for CIFAR-10 training set \( (n = 59000, d = 960, c = 32) \) corresponding to (a) projected data \( V \) after basic PCA, (b) rotated and projected data \( RV \) with ITQ, and (c) with our algorithm uniformizing the diagonal. One can note that diagonal coefficients for ITQ tend to be identical. So it makes sense to uniformize the diagonal.

**Lemma 1.** If \( R \) is an orthogonal matrix i.e. \( R R^T = R^T R = I \) where \( I \) stands for the identity matrix and \( M \) is a symmetric matrix, then \( \text{Tr}(R M R^T) = \text{Tr}(M) \). In particular \( R \) can be a rotation.

According to Lemma 1, since \( \text{Cov} \) is symmetric, \( \text{Tr}(R \text{Cov} VR^T) = \text{Tr}(\text{Cov}) \). So in order to have \( R \text{Cov} V R^T \) with equal diagonal coefficients, we should set \( \tau = \text{Tr}(\text{Cov})/c \).

So, similarly to IsoHash from work in (Kong and Li [2012]), we formulate the problem of finding \( R \) as the problem of equalizing the diagonal coefficients of \( \text{Cov} \) to the value \( \tau = \text{Tr}(\text{Cov})/c \).

3.2.1 Variance uniformization

We propose to build the optimal orthogonal matrix \( R \) as a product of \( c - 1 \) Givens rotations \( G(i, j, \theta) \) described by definition 3.1.

\(^{1}\)In the following, we will use equally the term orthogonal matrix or rotation.

\(^{2}\)If \( W \) is exactly the \( c \) first eigenvectors of \( C \) -for instance, if \( W \) is obtained through PCA-, \( \forall i \in \{1,...,c\} \), \( \sigma_i^2 = \lambda_i \).
Figure 1: For CIFAR-10 training set (\(n = 59000, d = 960, c = 32\)), covariance matrices of (a) projected data \(V\) after basic PCA, (b) rotated and projected data \(RV\) with ITQ, (c) with our algorithm.

**Definition 3.1.** A Givens rotation \(G(i, j, \theta)\) is a matrix of the form:

\[
G(i, j, \theta) = \begin{bmatrix}
1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
0 & \cdots & c & \cdots & -s & \cdots & 0 \\
0 & \cdots & \ddots & \cdots & \vdots & \cdots & \vdots \\
0 & \cdots & \vdots & \cdots & c & \cdots & 0 \\
0 & \cdots & \vdots & \cdots & \ddots & \cdots & \vdots \\
0 & \cdots & \vdots & \cdots & \vdots & \cdots & 1
\end{bmatrix}
\]

where \(c = \cos(\theta)\) and \(s = \sin(\theta)\) are at the intersections of the \(i\)-th and \(j\)-th rows and columns. The nonzero elements are consequently: \(\forall k \neq i, j, g_{k,k} = 1\), \(g_{i,i} = g_{j,j} = c\), \(g_{j,i} = -s\) and \(g_{i,j} = s\) for \(i > j\).

The computation of \(R\) follows the iterative Jacobi eigenvalue algorithm known as diagonalization process (Golub and van der Vorst [2000]):

\[
\text{CovV} \leftarrow G(i, j, \theta_t) \text{CovV} G(i, j, \theta_t)^T \\
R \leftarrow R G(i, j, \theta)^T.
\]

Note that left (resp. right) multiplication by \(G(i, j, \theta)\) only mixes \(i\)-th and \(j\)-th rows (resp. columns). The update from equation 1 only modifies \(i\)-th and \(j\)-th rows and columns of \(\text{CovV}\). The two updated diagonal coefficients \((i,i)\) and \((j,j)\) only depend on \(\text{CovV}_{ij}\), \(\text{CovV}_{ji}\), \(\text{CovV}_{ii}\) and \(\text{CovV}_{jj}\) and \(\theta\) which reduces the optimization of \(\theta\) to a 2-dimensional problem, a classical trick when using Givens rotations (Golub and van der Vorst [2000]). Let us define:

\[
a \overset{\text{def}}{=} \text{CovV}_{ij}, \ b \overset{\text{def}}{=} \text{CovV}_{ii}, \ c \overset{\text{def}}{=} \text{CovV}_{ji} = \text{CovV}_{ij}
\]

and:

\[
(a', b') \overset{\text{def}}{=} \begin{pmatrix} c & -s \\ s & c \end{pmatrix}, \quad (a, b) \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}.
\]

then we have the following result:

**Theorem 2.** If \(\min(a,d) \leq \tau \leq \max(a,d)\) (sufficient condition\(^3\)) then there exists one \(\theta \in [-\pi/2; \pi/2]\) such that \(a' = \tau\), \(d' = a + d - \tau\) and \(b' = -s_2 \sqrt{(a-d)^2 + b^2}\) with \(\cos(\theta) = \sqrt{\frac{1+c^2}{1+c^2-s_2^2}}\) and \(\sin(\theta) = \quad \).
\[
-\frac{c_1 s_2 + c_2 s_1}{2 \cos \theta}, \quad c_1 = \frac{a + d}{2} / \sqrt{\left(\frac{a - d}{2}\right)^2 + b^2}, \quad s_1 = b / \sqrt{\left(\frac{a - d}{2}\right)^2 + b^2}, \quad c_2 = (\tau - \frac{a + d}{2}) / \sqrt{\left(\frac{a - d}{2}\right)^2 + b^2} \text{ and } s_2 = \sqrt{1 - c_2^2} \in \mathbb{R}^+.
\]

**Proof.** See the proof in appendix.

Note that there is no need to compute explicitly \( \theta, \theta_1 \) or \( \theta_2 \).

### 3.2.2 Description of the diagonal uniformization algorithm

The mean of \( \text{CovV} \) diagonal coefficients being equal to \( \tau \), the following indices sets are not empty:

**Definition 3.2.** Let \( i\text{Inf} \stackrel{\text{def}}{=} \{ l \in \{1, \ldots, c\} \mid \text{CovV}_{l,l} < \tau \} \) and \( i\text{Sup} \stackrel{\text{def}}{=} \{ l \in \{1, \ldots, c\} \mid \text{CovV}_{l,l} > \tau \} \).

Taking one index \( j \) from \( i\text{Inf} \) and the other one \( i \) from \( i\text{Sup} \) guarantees the condition of Theorem 2 which allows to set \( \text{CovV}_{j,j} \) to the value \( \tau \). The index \( j \) can then be removed from \( i\text{Inf} \) and as \( \text{CovV}_{i,i} \) is set to \( a + d - \tau \), the index \( i \) reassigned to \( i\text{Inf} \) if \( \tau > \frac{a + d}{2} \) or in \( i\text{Sup} \) if \( \tau < \frac{a + d}{2} \). The number of diagonal coefficients of \( \text{CovV} \) different from \( \tau \) has been decreased by one. Finally, the necessary number of iterations to completely empty \( i\text{Inf} \) and \( i\text{Sup} \), i.e., uniformizing \( \text{CovV} \) diagonal, is bounded by \( c - 1 \). The method is summarized in algorithm 1 where \( \text{pop(list)} \) and \( \text{add(list, c)} \) are subroutines to delete and return the first element of \( \text{list} \), resp. to add \( c \) in \( \text{list} \).

**Algorithm 1** Diagonal Uniformization algorithm (UnifDiag)

1: Inputs : the \( c \times c \) symmetric matrix to uniformize : \( \text{CovV} \), tolerance : \( \text{tol} \)
2: // Initialization:
3: \( R \leftarrow I_c \) // Identity matrix with dimension \( c \)
4: \( \tau \leftarrow \text{Tr}(\text{CovV}) / c \); \( \text{nIter} = 0 \)
5: \( i\text{Inf} = \{ l \in \{1, \ldots, c\} \mid \text{CovV}_{l,l} < \tau - \text{tol} \} \)
6: \( i\text{Sup} = \{ l \in \{1, \ldots, c\} \mid \text{CovV}_{l,l} > \tau + \text{tol} \} \)
7: while \( \text{nIter} < c - 1 \) & not isEmpty(iInf) & not isEmpty(iSup) do
8: // Givens rotation parameters computation:
9: \( j \leftarrow \text{pop(iInf)} \); \( i \leftarrow \text{pop(iSup)} \); \( a \leftarrow \text{CovV}_{j,j} \); \( b \leftarrow \text{CovV}_{i,j} \); \( d \leftarrow \text{CovV}_{i,i} \);
10: \( c, s \) determined in Theorem 2
11: // CovV update:
12: \( \text{row}_j \leftarrow \text{CovV}_{j,:} \); \( \text{row}_i \leftarrow \text{CovV}_{i,:} \)
13: \( \text{CovV}_{j,:} = c \times \text{row}_j - s \times \text{row}_i \); \( \text{CovV}_{i,:} = s \times \text{row}_j + c \times \text{row}_i \)
14: \( \text{CovV}_{[i,j]} = \text{CovV}_{[j,i]} \); \( \text{CovV}_{[j,i]} = \text{CovV}_{[i,j]} \)
15: \( \text{CovV}_{[j,j]} = a' \); \( \text{CovV}_{[i,i]} = d' \); \( \text{CovV}_{[j,j]} = b' \) determined in Theorem 2
16: // Rotation update:
17: \( \text{col}_j \leftarrow R[:,j] \); \( \text{col}_i \leftarrow R[:,i] \)
18: \( R[:,j] = c \times \text{col}_j - s \times \text{col}_i \); \( R[:,i] = s \times \text{col}_j + c \times \text{col}_i \)
19: // Indices list update:
20: if \( \frac{a + d}{2} < \tau - \text{tol} \) then
21: \( \text{add(iInf, i)} \)
22: end if
23: if \( \frac{a + d}{2} > \tau + \text{tol} \) then
24: \( \text{add(iSup, i)} \)
25: end if
26: end while
27: return \( R \)
3.2.3 Relation to existing works regarding complexities

Our algorithm for computing $R$ requires the storage of only the $c \times c \text{CovV}$ matrix. At most $c - 1$ Givens rotations are computed, each implying four column or row multiplications i.e. $4c$ flops. So the final time complexity is $O(c^3)$.

**Spatial complexity**  ITQ requires the storage of two $c \times n$ matrices: the after-PCA projected dataset $V$ and the corresponding binary encoding matrix $B$. As ITQ is dependent on the number of training data, it scales poorly w.r.t the size of the datasets and is of limited use for larger-than-memory datasets.

**Time complexity**  Learning $R$ with our model is computationally cheaper than taking a random rotation as it implies a QR decomposition with a cost of $O(c^3)$.

This is also the case for ITQ where for a certain number of iterations, the $c \times c$ product $B^T V$ should be computed ($O(c^2 n)$), followed by an SVD ($O(c^3)$) corresponding to a final cost: $O(K c^2 n + K c^3)$ where $K$ is the number of iterations and set in practice to 50. Moreover, one can remark that in the proposed model, the computation time for $R$ does not depend on the size of the training set which makes it scalable.

In IsoHash ([Kong and Li, 2012]), authors have already formulated the problem as a variance-uniformization one. Two algorithms were proposed: Lift and projection and Gradient Flow, both looking for an orthogonal matrix $R$ corresponding to a matrix with a uniform diagonal and the same eigenvalues as CovV. Lift and projection (LP) consists in alternative projections on the space of matrices with uniform diagonal and the space of matrices with the same eigenvalues as CovV. This costs $n_{iter} \text{ SVD of a } c \times c$ matrix. So the corresponding time complexity is $O(c^3 \times n_{iter})$ with $n_{iter}$ in practice set to 100. Gradient Flow (GF) involves an integration of a differential equation using Adams-Bashforth-Moulton PECE solver, with a cost equal to $O(c^3)$. So LP is very slow while GF barely faster. In both cases, even if $c$ is small in comparison to $d$ and the complexities do not either depend on $n$, our model has the great advantages 1) to have a lower time complexity cost, 2) to be much more simpler than IsoHash.

4 The proposed streaming unsupervised model for binary quantization

In the streaming context, the goal is to have $b_t = \text{sign}(\tilde{W}_t x_t)$ for $t = 1...n$ where $b_t$ is computed and returned before $x_{t+1}$ is seen by using $x_t$ and previous update of $W_{t-1}$. $\tilde{W}_t = R_t W_t$ where $W_t$ is the linear dimension reduction embedding and $R_t$ is a suitable $c \times c$ rotation.

**Online update of the principal subspace**  Fast Orthonormal PAST (Projection Approximation and Subspace Tracking) ([Abed-Meraim et al., 2000]), also named OPAST, is a method to quickly estimate and track the principal subspace of a data stream, corresponding to matrix $W$. At each iteration $t$, OPAST guarantees the orthonormality of $W_t$ rows and costs only $4dc + O(c^2)$ flops while storing only $W$ and a $c \times c$ matrix. The pseudo-code is given by algorithm [3] in appendix.

**Online update of a suitable rotation**  The procedure to compute $R_t$ relies exclusively on the $c \times c$ covariance matrix of projected data $\text{CovV}_t$ and not on the whole $V$ as for ITQ. $\text{CovV}_t$ is easy to update dynamically and this is performed while computing $W_t$ with OPAST algorithm, so no need of adaptation for the $R_t$ learning in the streaming setting.

Remark also that this stage is completely independent from the first one to compute the principal subspace. One interest of UnifDiag algorithm for learning the rotation is the freedom to plug any other method for online PCA ([Feng et al., 2013], [Yang and Xu, 2015]) or online subspace tracking ([Abed-Meraim et al., 2000]).
4.1 Discussion about the number of passes over data

A crucial point is to determine at which stage data should be projected and rotated. Should we perform a first pass over the data in order to work on a reliable \( W_t \)?

One pass over the data will give poorer performance relatively to the other approaches with two or three passes since data are directly projected onto \( W_t \) which is not properly estimated during the first iterations. Consequently, \( R_t \) is also learned from an approximate estimation of projected data covariance at step \( t \): \( \text{Cov}V_t \).

If three passes are possible over the data, results will be optimal and far more better than only one pass. One pass can be used for obtaining \( W \) and a second for \( \text{Cov}V \). Then, \( R \) can be correctly computed and a third pass is used for getting the final binary embedding: \( x_i \) is projected onto \( R_t W_t \) and the function sign is applied. This is still a streaming algorithm which reduces the memory storage requirements in comparison to ITQ and there is no loss of accuracy.

Alternatively, two passes can be done: one for \( W \) and \( \text{Cov}V \) and a second for the encoding -rotation and sign applications- leading to results with intermediate quality in comparison with the one and three passes approaches.

4.2 Complexity analysis

We compare here the spatial and time complexities of our method and OSH \cite{Leng2015} which is to the best of our knowledge, the only online hashing method the most similar to ours, i.e. unsupervised, hyperplanes-based and reading one data point at a time. Despite what is announced, OSH is fundamentally mini-batch: the stream is divided into chunks of data for which a matrix \( S \in \mathbb{R}^{d \times l} \) as a sketch of the whole dataset \( X \in \mathbb{R}^{d \times n} \) is maintained. Then the principal components are computed from the updated sketch \( S \). The projection of data followed by the random rotation can be applied only after this step. Therefore there are actually two passes over the data by reading twice data of each chunk.

**Spatial complexity** Without counting the projection matrix and the rotation, OSH needs spatially to maintaining the sketch \( S \) which costs \( O(d \times l) \) with \( l \ll d \). The SVD decomposition then needs \( O(ld + l^2) \). In comparison, we only need \( O(c^2) \) with \( c \ll l \).

**Time complexity** For each round, OSH takes \( O(dl^2 + l^3) \) to learn the principal components. If we consider only one pass over the data, as new data is seen, our method updates with OPAST the estimation of the principal subspace and the covariance of the projected data \( \text{Cov}V \) with only a cost of \( 4dc + O(c^2) \). The complexity to compute the rotation is \( O(c^2) \).

Our method shows clearly advantages in terms of spatial and time complexities. Moreover, binary hash codes can be directly computed as new data is seen, while OSH, as a mini-batch method, has a delay.

5 Experiments

All experiments were made with the CIFAR-10 dataset\(^4\) which contains 60000 32 \times 32 color images equally divided in 10 classes. 960-dimensional GIST descriptors were extracted from those data.

5.1 Nearest neighbor search task - Comparison to offline methods

We evaluate the performance of nearest neighbor search using Euclidean neighbors as a ground truth. A nominal threshold of the average distance to the 50th nearest neighbor is computed and determines the sets of neighbors and non-neighbors called Euclidean ground truth. Based on that, the precision-recall curve is plotted for different hashing methods with the following setting: 1000 queries were randomly

\(^4\)http://www.cs.toronto.edu/~kriz/cifar.html
sampled and the remaining data is used as training set. All the corresponding experiments are averaged over 5 random training/test partitions.

We compared our method to four baseline methods that follow the basic hashing scheme $H(X) = \text{sgn}(\tilde{W}X)$, where the projection matrix $\tilde{W} \in \mathbb{R}^{c \times d}$ is determined according to the chosen method:

- **LSH** (Andoni and Indyk [2008]): $\tilde{W}$ is a Gaussian random matrix.
- **basic-PCA**: $\tilde{W}$ whose rows are the $c$ principal components, supposed to perform badly compared to the next method with a random rotation.
- **RandRot-PCA**: $\tilde{W} = RW$ where $W$ is the PCA matrix and $R$ a random orthogonal (rotation) matrix. This is the initialization of ITQ.
- **RotUnifDiagOffline-PCA**: $\tilde{W} = RW$ where $W$ is the PCA matrix and $R$ the rotation obtained with UnifDiag.
- **ITQ**: as described in part 3.
- **RandRot-OPAST**: $W$ is the PCA matrix obtained through one pass with OFAST and $R$ a random rotation. (2 passes)
- **RotUnifDiagOffline-OPAST**: $W$ is OFAST-principal subspace (1 pass) and $R$ is obtained with UnifDiag through another pass. (3 passes)
- **RotUnifDiagOnline-OPAST**: $W$ is OFAST-principal subspace (1 pass) and $R$ is computed during the same pass with UnifDiag. (2 passes)

Figure 2 shows the results for CIFAR-10 dataset, varying the length of the code. Because our algorithm cannot produce codes with a length higher than the original data dimension ($d = 960$), code sizes are evaluated up to 512 bits. For space constraints, results for $c = 256$ and $c = 512$ are presented in appendix. The bigger the area under the recall-precision curve, the better is the method. As expected, basic-PCA performs poorly for each $c$. Then, the random projection holds the second worst results for $c = 32$ and $c = 64$ while other methods have similar performance. We remark though, that ITQ and RotUnifDiagOffline-OPAST (our online algorithm with 3 passes) give the best results for all code sizes. As the code length gets higher, they are more closely followed by randRot-PCA, randRot-OPAST (2 passes) and RotUnifDiagOffline-PCA (our offline model). In particular, random projection does not give the second worst results from $c = 128$ and behaves similarly to the other methods, except RotUnifDiagOffline-PCA which is outperformed from $c = 128$. 

---

![Figure 2: Precision-Recall curves on CIFAR-dataset.](image-url)
5.2 Nearest neighbor search task - Comparison to OSH

In this section, we evaluate our online algorithm in comparison with Online Sketching Hashing (OSH) \cite{Lenc2015}, the nearest state-of-the-art approach. The training dataset is equally divided into 100 chunks (100 rounds) in order to perform a fair comparison since OSH is mini-batch and therefore performs two passes over the data: one to feed the chunk, an another one to perform the projection on $W_t$ (computed with an SVD on the sketched version of the data chunk) followed by the random rotation application. Figure 3 shows the Mean Average Precision (MAP) on CIFAR-10 dataset with 100 rounds for different value of $c < l = 200$ averaged on 5 experiments ($c = 16$ and $c = 128$ in appendix). Our algorithm outperforms OSH for $c \in \{16, 32, 64\}$. Finally, for $c = 32$, figure 4 illustrates the convergence property of our online algorithm by plotting the MAP evolution (averaged on 5 experiments) as new data is seen. The results are shown for the first 1000 data points of the training set: we do not need more than 400 points to reach the precision of the method.

6 Conclusion

In this paper we introduced a novel method for learning distance-preserving binary embeddings of high-dimensional data with convergence guarantees. Unlike classical state-of-the-art methods, our algorithm is straightforwardly adaptable to the streaming context where the whole dataset cannot be stored entirely. In the streaming context, our online model enables to obtain without delay a binary code as a new data point is seen. Our approach shows promising results for both offline and online settings, as evidenced by the experimental part. It can achieve accuracy at least similar to if not better than state-of-the-art methods while saving considerable computation time and spatial requirements. Further work would be to investigate whether another rotation, not uniformizing the diagonal of the covariance matrix of the projected data, could be more optimal. Another interesting perspective would be to evaluate the performance of the compact binary codes in other machine learning applications: instead of using the original data, one could use directly these binary embeddings to perform unsupervised or supervised learning hopefully without downgrading the accuracy.

References

K. Abed-Meraim, A. Chkeif, and Y. Hua. Fast orthonormal past algorithm. IEEE Signal Processing Letters, (3):60 – 62, 2000.

A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM, (1):117–122, 2008.
A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal LSH for angular distance. In *NIPS*, pages 1225–1233, 2015.

M. Bojarski, A. Choromanska, K. Choromanski, F. Fagan, C. Gouy-Pailler, A. Morvan, N. Sakr, T. Sarlos, and J. Atif. Structured adaptive and random spinners for fast machine learning computations. In *AISTATS*, pages 1020–1029, 2017.

W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *ICML*, pages 2285–2294, 2015.

J. Feng, H. Xu, and S. Yan. Online robust pca via stochastic optimization. In *NIPS*, pages 404–412, 2013.

G. H. Golub and H. A. van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, (1–2):35 – 65, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011.

Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (12):2916–2929, 2013.

K. Grauman and B. Kulis. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1092–1104, 2011.

L.-K. Huang, Q. Yang, and W.-S. Zheng. Online hashing. In *IJCAI*, pages 1422–1428, 2013.

H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, 2010.

W. Kong and W.-j. Li. Isotropic hashing. In *NIPS*, pages 1646–1654. 2012.

H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.

Y. Lee. Spherical hashing. In *CVPR*, pages 2957–2964, 2012.

C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu. Online sketching hashing. In *CVPR*, pages 2503–2511, 2015a.

C. Leng, J. Wu, J. Cheng, X. Zhang, and H. Lu. Hashing for distributed data. In *ICML*, pages 1642–1650, 2015b.

W. Liu, J. Wang, and S. fu Chang. Hashing with graphs. In *ICML*, 2011.

W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427. 2014.

M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517. 2009.

R. Raziperchikolaei and M. Á. Carreira-Perpiñán. Optimizing affinity-based binary hashing using auxiliary coordinates. In *NIPS*, pages 640–648, 2016.

K. Terasawa and Y. Tanaka. Spherical LSH for approximate nearest neighbor search on unit hypersphere. In *WADS*, pages 27–38, 2007.
J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, (12):2393–2406, 2012.

J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data - a survey. *Proceedings of the IEEE*, (1):34–57, 2016.

Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

W. Yang and H. Xu. Streaming sparse principal component analysis. In *ICML*, pages 494–503, 2015.

F. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *ICML*, 2014.
7 Appendix

7.1 ITQ algorithm

This section presents ITQ pseudo-code in algorithm 2. \textit{randomOrthogonalMatrix}(c,c) denotes a subroutine returning a \(c \times c\) random orthogonal matrix which simply generates a \(c \times c\) random matrix and then applies a QR decomposition with time complexity \(O(c^3)\).

\begin{algorithm}
\caption{ITQ algorithm [Gong et al. 2013]}
\begin{algorithmic}[1]
\STATE \textbf{Inputs}: data : \(V = WX, V \in \mathbb{R}^{c \times n}\) PCA-projected \(X\); code length: \(c\); number of iterations: \(K\)
\STATE \(R \leftarrow \text{randomOrthogonalMatrix}(c,c)\) // Initialization
\FOR {\(i\) in \(1 \ldots K\)}
\STATE \(B = \text{sign}(RV)\) // Fix \(R\) and update \(B\)
\STATE \(S, \Omega, \tilde{S}^T = \text{SVD}(V^T B)\) // Fix \(B\) and update \(R\)
\STATE \(R = \tilde{S} S^T\)
\ENDFOR
\end{algorithmic}
\end{algorithm}

7.2 Proof of theorem 2

In this section Theorem 2 is proven.

\textbf{Theorem 2} If \(\min(a,d) \leq \tau \leq \max(a,d)\) (sufficient condition\footnote{This theorem uses only a sufficient condition. A weaker necessary and sufficient condition to guarantee \(|c_2| < 1\) and \(s_2 \in \mathbb{R}^+\) is \(\frac{a+d}{2} - \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2} \leq \tau \leq \frac{a+d}{2} + \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}\) (proof by setting \(b = 0\)).}) then there exists one \(\theta \in \left[\frac{\pi}{2}; \frac{\pi}{2}\right]\) such that \(a' = \tau\) with \(\cos(\theta) = \sqrt{1+c_1s_2-s_1s_2^2}\), \(c_1 = \frac{a-d}{2}\sqrt{\frac{a-d}{2} + b^2}\), \(s_1 = \frac{b}{\sqrt{(a-d)^2 + b^2}}\), \(c_2 = \frac{(a-d)}{2}\sqrt{\frac{a-d}{2} + b^2}\) and \(s_2 = \sqrt{1-c_2^2} \in \mathbb{R}^+\).

\textit{Proof.} As stated in equation 3, the problem of finding a suitable angle \(\theta\) to uniformize the diagonal of \(\text{Cov}V\) can be reduced to the following 2-dimensional problem:

\begin{equation}
\begin{pmatrix}
a' \\
b'
c' \\
d'
\end{pmatrix}
def
\begin{pmatrix}
c & -s \\
s & c
\end{pmatrix}\begin{pmatrix}
a & b \\
b & d
\end{pmatrix}\begin{pmatrix}
c & s \\
-s & c
\end{pmatrix}
\end{equation}

where \(a \deff \text{Cov}V_{j,j}, d \deff \text{Cov}V_{i,i}, b \deff \text{Cov}V_{j,i} = \text{Cov}V_{i,j}\). This implies :

\begin{align}
a' &= \frac{a+d}{2} + \frac{(a-d)}{2} b \cdot \begin{pmatrix}
\cos(2\theta) \\
-\sin(2\theta)
\end{pmatrix} \\
d' &= \frac{a+d}{2} - \frac{(a-d)}{2} b \cdot \begin{pmatrix}
\cos(2\theta) \\
-\sin(2\theta)
\end{pmatrix} \\
b' &= \frac{(a-d)}{2} b \cdot \begin{pmatrix}
\sin(2\theta) \\
\cos(2\theta)
\end{pmatrix}
\end{align}

As, the Givens angle \(\theta\) should be parameterized so that the diagonal coefficients are set to a same value \(\tau\), the following holds:

\begin{equation}
\begin{pmatrix}
\frac{a-d}{2} \\
\frac{b}{2}
\end{pmatrix} \cdot \begin{pmatrix}
\cos(2\theta) \\
-\sin(2\theta)
\end{pmatrix} = \tau - \frac{a+d}{2}
\end{equation}
Recall that:

\[
\begin{align*}
    c_1 & \overset{\text{def}}{=} \cos(\theta_1) = \frac{a - d}{2} / \sqrt{\frac{(a - d)^2}{4} + b^2} \\
    s_1 & \overset{\text{def}}{=} \sin(\theta_1) = b / \sqrt{\frac{(a - d)^2}{4} + b^2} \\
    c_2 & \overset{\text{def}}{=} \cos(\theta_2) = \left( \frac{\tau - a + d}{2} \right) / \sqrt{\frac{(a - d)^2}{4} + b^2} \\
    s_2 & \overset{\text{def}}{=} \sin(\theta_2) = \sqrt{1 - c_2^2}.
\end{align*}
\]

Remark that the condition \( \min(a, d) < \tau < \max(a, d) \) guarantees \( c_2 \) to be well defined i.e. \( |c_2| \leq 1 \) and \( s_2 \in \mathbb{R}^+ \). Then, equation 8 becomes:

\[
\begin{pmatrix}
    c_1 \\
    s_1
\end{pmatrix}
\begin{pmatrix}
    \cos(2\theta) \\
    -\sin(2\theta)
\end{pmatrix}
= c_2.
\]

This is clear that a solution of equation 13 is:

\[
\begin{pmatrix}
    \cos(2\theta) \\
    -\sin(2\theta)
\end{pmatrix}
= \begin{pmatrix}
    c_1 c_2 - s_1 s_2 \\
    c_1 s_2 + c_2 s_1
\end{pmatrix}.
\]

In that case, one can take:

\[
\begin{align*}
    \cos(\theta) &= \sqrt{\frac{1 + \cos(2\theta)}{2}} = \sqrt{\frac{1 + c_1 c_2 - s_1 s_2}{2}} \\
    \sin(\theta) &= \frac{\sin(2\theta)}{2 \cos(\theta)} = -\frac{c_1 s_2 + c_2 s_1}{2 \cos \theta}
\end{align*}
\]

and the corresponding Givens rotation gives:

\[
\begin{align*}
    a' &= \tau \\
    d' &= a + d - \tau \\
    b' &= \left( \frac{a - d}{2} \right) \begin{pmatrix}
        \sin(2\theta) \\
        \cos(2\theta)
    \end{pmatrix} \\
    &= \sqrt{\left( \frac{a - d}{2} \right)^2 + b^2} \begin{pmatrix}
        c_1 \\
        s_1
    \end{pmatrix} \begin{pmatrix}
        c_1 c_2 - s_1 s_2 \\
        c_1 s_2 + c_2 s_1
    \end{pmatrix} \\
    &= -s_2 \sqrt{\left( \frac{a - d}{2} \right)^2 + b^2}
\end{align*}
\]

with \( s_2 \) completely defined by equations 12 and 11.

### 7.3 OPAST algorithm

The pseudo-code of our modified OPAST algorithm tracking online the principal subspace and the covariance matrix \( \text{CovV} \) of projected data onto this principal subspace is given here.

### 7.4 Further experiments

Experiments have been carried out on a single processor machine (Intel Core i7-5600U CPU @ 2.60GHz, 4 hyper-threads) with 16GB RAM. In this section, we present other experimental results.
Algorithm 3 Modified OPAST algorithm

1: **Inputs**: stream : \( x_1, x_2, \ldots x_n \in \mathbb{R}^d \), dimension of the principal subspace \( c \), forgetting factor \( \alpha \in (0, 1] \)

2: // Initialization:
3: \( W \leftarrow \text{randomOrthonormalMatrix}(d, c) ; \quad \text{CovV} \leftarrow 0_c \)
4: for \( x_t \) in stream do
5: \( y := W^T x ; \quad \text{CovV} \leftarrow \alpha \text{CovV} + y.y^T ; \quad q := \frac{1}{\alpha} Z y ; \quad \gamma := \frac{1}{1 + y^T q} ; \quad p := \gamma (x - Wy) \)
6: \( Z \leftarrow \frac{1}{\alpha} Z - \gamma q q^T \)
7: \( \tau := \frac{1}{||q||^2} \left( \frac{1}{\sqrt{1 + ||p||^2 ||q||^2}} - 1 \right) ; \quad p' := \tau W.q + (1 + \tau ||q||^2) p ; \quad W \leftarrow W + p' q^T \)
8: end for
9: return \( W, \text{CovV} \)

| \( c \)  | 32  | 64  | 128 | 256 | 512 |
|--------|-----|-----|-----|-----|-----|
| ITQ    | 1.8 | 3.6 | 9.3 | 25.8| 87.6|
| UnifDiag | 3 \times 10^{-3} | 5 \times 10^{-3} | 1 \times 10^{-2} | 4 \times 10^{-2} | 0.14 |

7.5 Speedups with rotation computation in comparison with ITQ in offline setting

Table 1 shows the speedup results regarding the rotation time computation with our method \( \text{UnifDiag} \) based on uniformizing the diagonal of \( \text{CovV} \) (\( \text{tol} = 10^{-14} \)) instead of ITQ (number of iterations set to \( K = 50 \)). This empirically confirms that our method for computing \( R \) is faster than ITQ (see Section 3.2.3 for exact theoretical time complexities) and is a very interesting alternative to ITQ in the offline setting.

7.5.1 Nearest neighbor search task in the offline setting

Figure 5 shows the complete results with CIFAR-10 dataset for the precision-recall curves, varying the length of the code from \( c = 32 \) to \( c = 512 \). Table 2 contains the corresponding Mean Average Precision values. Both emphasize the competitive results of our method in comparison with ITQ.

7.5.2 Nearest neighbor search task in the online setting

Figure 6 shows the complete online MAP results on CIFAR-10 dataset from \( c = 32 \) to \( c = 128 \).
Figure 5: Precision-Recall curves on CIFAR-dataset.

Table 2: Mean Average Precision for offline methods

| method / c   | 32  | 64  | 128 | 256 | 512 |
|--------------|-----|-----|-----|-----|-----|
| RotUnifDiagOffline-OPAST | 0.326 | 0.368 | 0.398 | 0.405 | 0.439 |
| random projection | 0.275 | 0.325 | 0.362 | 0.388 | 0.435 |
| basic-PCA | 0.259 | 0.278 | 0.282 | 0.266 | 0.277 |
| RotUnifDiagOffline-PCA | 0.320 | 0.353 | 0.374 | 0.360 | 0.377 |
| RotUnifDiagOnline-OPAST | 0.325 | 0.371 | 0.399 | 0.406 | 0.441 |
| randRot-PCA | 0.322 | 0.368 | 0.400 | 0.411 | 0.448 |
| randRot-OPAST | 0.320 | 0.365 | 0.400 | 0.409 | 0.448 |
| ITQ | **0.329** | **0.375** | **0.403** | 0.405 | 0.441 |
Figure 6: Mean Average Precision on CIFAR-10 dataset.

(a) $c = 16$

(b) $c = 32$

(c) $c = 64$

(d) $c = 128$