Streaming Binary Sketching Based on Subspace Tracking and Diagonal Uniformization

Anne Morvan, Antoine Souloumiac, Cedric Gouy-Pailler, Jamal Atif

STREAMING BINARY SKETCHING BASED ON SUBSPACE TRACKING AND DIAGONAL UNIFORMIZATION

Anne Morvan[⋆†] *Antoine Souloumiac*[⋆] *Cédric Gouy-Pailler*[⋆] *Jamal Atif*[†]

[⋆] CEA, LIST, 91191 Gif-sur-Yvette, France
[†]Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, 75016 Paris, France

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 online 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 c-bits binary code for preserving the distance between the points from the original high-dimensional space. Our algorithm does not require neither the storage of the whole dataset nor a chunk, thus it is fully adaptable to the streaming setting. It also provides low time complexity and convergence guarantees. We demonstrate the quality of our binary sketches through experiments on real data for the nearest neighbors search task in the online setting.

***Index Terms*—** Sketching, streaming, subspace tracking, Givens rotations

1. INTRODUCTION

In large-scale machine learning applications such as computer vision or metagenomics, learning similarity-preserving binary codes is critical 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) [1] for nearest neighbors search: 1) input data with high dimension d are projected onto a lower c-dimensional space through a $c \times d$ random projection with i.i.d. Gaussian entries, 2) then a hashing function is applied to the resulting projected vectors to obtain the final binary codes. Two examples for the hashing function are cross-polytope LSH [2] which returns the closest vector from the set $\{\pm 1 e_i\}_{1 \leq i \leq c}$ where $\{e_i\}_{1 \leq i \leq c}$ stands for the canonical basis, and hyperplane LSH [3], the sign function applied pointwise. To reduce the storage cost of the projection matrix and the matrix-vector products computation times ($O(c \times d)$), a structured pseudo-random matrix can be used instead [3, 4] with a reduced time complexity of $O(d \log c)$ thanks to fast Hadamard and Fourier transforms.

In the context of nearest neighbors search or classification, the accuracy of the data sketches can be improved by learning this projection from data [5, 6, 7, 8, 9]. As Principal Component Analysis (PCA) is a common tool for reducing data dimensionality, data are often projected onto the first c principal components. But PCA alone is not sufficient. Indeed, the c first principal components are chosen with a decreasing order of explained variance: principal directions with higher variance carry more information. Thus, 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: it balances variance over the principal components. In work from [10], a random rotation is used giving quite good results. In ITerative Quantization (ITQ) [7] or in Isotropic Hashing (IsoHash) [8], the rotation is rather learned. In ITQ, the rotation is iteratively computed by solving an orthogonal Procustes problem: the alternating minimization of the quantization error of mapping data to the vertices of the 2^c hypercube. This technique is currently the state-of-the-art for computing similarity-preserving binary codes based on PCA. ITQ and IsoHash come with a major drawback, though. They are completely offline since the whole dataset needs to be stored for computing the c principal components. This can be prohibitive when dealing with lots of high-dimensional data.

Contributions: We introduce a streaming algorithm with convergence guarantees where data is seen only once and principal subspace plus the balancing rotation are updated as new data is seen. To obtain the principal subspace and the rotation, this requires additionally only the storage of two $c \times c$ matrices, instead of the whole initial and projected datasets as for ITQ or IsoHash. Our algorithm outperforms the known state-of-the-art online unsupervised method Online Sketching Hashing (OSH) [11] while being far less computationally demanding.

A. Morvan is partly supported by the DGA (French Ministry of Defense).

2. RELATED WORK

Two paradigms exist to build hash functions [12]: data-independent [1, 13, 14] and data-dependent methods. The latter ones learn the hash codes from a training set and perform better. The learning can be unsupervised [5, 15, 7, 8, 16, 17, 9, 18] aiming at preserving distances in the original space or (semi-)supervised which also tries to preserve label similarity [6, 19]. Some recent hashing functions involve deep learning [20, 21]. When the dataset is too large to be loaded into memory, distributed [22] and online hashing techniques [23, 11, 24] have been developed. Online Hashing (OKH) [23] learns the hash functions from a stream of similarity-labeled pair of data with a "Passive-Aggressive" method. Recent approach from [24] relies on Mutual Information. Although claimed to be unsupervised, it is supervised as similarity labels are also needed between pairs of data. In Online Sketching Hashing (OSH) [11], 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 ONLINE UNSUPERVISED MODEL FOR BINARY QUANTIZATION

3.1. Notations and problem statement

We have a stream of n data points $\{x_t \in \mathbb{R}^d\}_{1\leq t\leq n}$ supposed to be zero-centered. The goal is to have $b_t = \text{sign}(\tilde{W}_t x_t) \in \{-1, 1\}^c$ for $t = 1 \dots n$ where c denotes the code length, $c \ll d$, s.t. for each bit $k = 1 \dots c$, the binary encoding function is defined by $h_k(x_t) = \text{sign}(\tilde{w}_{k,t}^T x_t)$ where $\tilde{w}_{k,t}$ 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. So $\tilde{w}_{k,t}^T$ is a row of $\tilde{W}_t$ for each k, with $\tilde{W}_t \in \mathbb{R}^{c\times d}$. b_t is computed and returned before x_{t+1} is seen by using x_t and $\tilde{W}_{t-1}$ solely. We consider in this paper the family of hash functions $\tilde{W}_t$ s.t. $\tilde{W}_t = R_t W_t$ where W_t is the linear dimension reduction embedding applied to data and R_t is a suitable $c \times c$ orthogonal matrix. Here, we take W_t as the matrix whose row vectors $w_{k,t}^T$ are the c first principal components of the covariance matrix $\Sigma_{X,t} = X_t X_t^T$ where X_t denotes the dataset seen until data t. So the challenge is in tracking the principal subspace in an online fashion as new data is seen and in defining an appropriate orthogonal matrix[1] to rotate the projected data onto this principal subspace.

Works from [7, 10] argued that defining an orthogonal transformation applied to the PCA-projected data which tends to balance the variances over the PCA-directions improves the quality of the hashing but R was first learned to uniformize the variances for different directions only in work from [8] with Isotropic Hashing (IsoHash). In the latter, the problem is explicitly described as learning a rotation which produces isotropic variances of the PCA-projected dimensions. We propose here a new simpler online method UnifDiag for learning a rotation to uniformize the diagonal of the covariance of projected data after tracking the principal subspace W_t from the data stream with Fast Orthonormal PAST (Projection Approximation and Subspace Tracking) [25], also named OPAST. At each iteration t, OPAST guarantees the orthonormality of W_t rows and costs only $4dc + O(c^2)$ flops while storing only W_t and a $c \times c$ matrix. The rotation learning is completely independent from the principal subspace tracking. Indeed, any other method for online PCA [26, 27] or subspace tracking [25] can be plugged before UnifDiag.

[1] In the sequel, we use equally the term orthogonal matrix or rotation.

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

Let the $c \times c$ symmetric matrix $\Sigma_{V,t} = V_t V_t^T$ be the covariance matrix of projected data seen until data t $V_t = W_t X_t$. R_t is learned for each t to balance the variance over the c directions given by the c principal components of $\Sigma_{V,t}$. $\Sigma_{V,t}$ is easy to update dynamically and we adapt OPAST algorithm to perform this while computing W_t. In the sequel, for clarity we drop the subscript t. Let us consider the c diagonal coefficients of Σ_V: $\sigma_1^2, ..., \sigma_c^2$ s.t. $\sigma_1^2 \geq ... \geq \sigma_c^2$. As Σ_V is symmetric, $\text{Tr}(\Sigma_V) = \sum_{i=1}^c \sigma_i^2 = \sum_{i=1}^c \lambda_i$ where Tr stands for the Trace application and $\lambda_1, ..., \lambda_c$ are the c first eigenvalues of Σ_X s.t. $\lambda_1 \geq ... \geq \lambda_c$[2]. Balancing variance over the c directions can be seen as equalizing the diagonal coefficients of Σ_V s.t. $\sigma_1^2 = ... = \sigma_c^2 \stackrel{def}{=} \tau$. Since Σ_V is symmetric, $\text{Tr}(R\,\Sigma_V R^T) = \text{Tr}(\Sigma_V)$. So in order to have $R\,\Sigma_V R^T$ with equal diagonal coefficients, we should set $\tau = \text{Tr}(\Sigma_V)/c$. So, similarly to IsoHash, we formulate the problem of finding R as the problem of equalizing the diagonal coefficients of Σ_V to the value $\tau = \text{Tr}(\Sigma_V)/c$. Our proposed optimal orthogonal matrix R is built as a product of $c-1$ Givens rotations $G(i, j, \theta)$ described by Def. 3.1.

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

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

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 $i > j$.

[2] If W is exactly the c first eigenvectors of Σ_X – for instance, if W is obtained through PCA –, then $\forall i \in \{1, ..., c\}$, $\sigma_i^2 = \lambda_i$.

The computation of R follows the iterative Jacobi eigenvalue algorithm known as diagonalization process [28]:

$$\Sigma_V \leftarrow G(i,j,\theta)\, \Sigma_V\, G(i,j,\theta)^T \quad (1)$$

$$R \leftarrow R\, G(i,j,\theta)^T. \quad (2)$$

Note that left (resp. right) multiplication by $G(i,j,\theta)$ only mixes i-th and j-th rows (resp. columns). The update from Eq. 1 only modifies i-th and j-th rows and columns of Σ_V. The two updated diagonal coefficients (i,i) and (j,j) only depend on $\Sigma_{V,i,i}$, $\Sigma_{V,j,j}$, $\Sigma_{V,j,i}$ and θ which reduces the optimization of θ to a 2-dimensional problem, a classical trick when using Givens rotations [28]. Then we have Th. 3.1 by defining: $a \stackrel{def}{=} \Sigma_{V,j,j}$, $d \stackrel{def}{=} \Sigma_{V,i,i}$, $b \stackrel{def}{=} \Sigma_{V,j,i} = \Sigma_{V,i,j}$,

$$\begin{pmatrix} a' & b' \\ b' & d' \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} a & b \\ b & d \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}. \quad (3)$$

Theorem 3.1 *If* $\min(a,d) \leq \tau \leq \max(a,d)$ *(sufficient condition)*[3] *then there exists one* $\theta \in [-\pi/2, \pi/2]$ *s.t.* $a' = \tau$, $d' = a + d - \tau$ *and* $b' = -s_2\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$ *with* $\cos(\theta) = \sqrt{\frac{1+c_1c_2-s_1s_2}{2}}$ *and* $\sin(\theta) = -\frac{c_1s_2+c_2s_1}{2\cos\theta}$, $c_1 = \left(\frac{a-d}{2}\right)/\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$, $s_1 = b/\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$, $c_2 = (\tau - \frac{a+d}{2})/\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$ *and* $s_2 = \sqrt{1-c_2^2} \in [0,1]$.

Proof 3.1 *With* $u(\theta) \stackrel{def}{=} \begin{pmatrix}\cos(2\theta) & -\sin(2\theta)\end{pmatrix}^T$, *Eq. 3 gives:* $a' = \frac{a+d}{2} + \begin{pmatrix}\frac{a-d}{2} & b\end{pmatrix}.u(\theta)^T$, $d' = \frac{a+d}{2} - \begin{pmatrix}\frac{a-d}{2} & b\end{pmatrix}.u(\theta)^T$ *and*

$b' = \begin{pmatrix}\frac{a-d}{2} & b\end{pmatrix}.\begin{pmatrix}\sin(2\theta)\\ \cos(2\theta)\end{pmatrix}$. *As the Givens angle* θ *should be parameterized s.t. all diagonal coefficients are set to* τ, $\begin{pmatrix}\frac{a-d}{2} & b\end{pmatrix}.u(\theta)^T = \tau - \frac{a+d}{2}$. *Previously defined* c_1, c_2, s_1, s_2 *and the condition* $\min(a,d) < \tau < \max(a,d)$ *gives* $|c_2| \leq 1$ *and* $s_2 \in [0,1]$. *Then the last equation becomes:* $\begin{pmatrix}c_1 & s_1\end{pmatrix}.u(\theta)^T = c_2$. *A clear solution is:* $u(\theta)^T = \begin{pmatrix}c_1c_2 - s_1s_2\\ c_1s_2 + c_2s_1\end{pmatrix}$. *Thus, one can take:* $\cos(\theta) = \sqrt{\frac{1+\cos(2\theta)}{2}} = \sqrt{\frac{1+c_1c_2-s_1s_2}{2}}$, $\sin(\theta) = \frac{\sin(2\theta)}{2\cos(\theta)} = -\frac{c_1s_2+c_2s_1}{2\cos\theta}$ *and the associated Givens rotation gives:* $a' = \tau$,

$$d' = a + d - \tau,\ \ b' = \begin{pmatrix}\frac{a-d}{2} & b\end{pmatrix}\begin{pmatrix}\sin(2\theta)\\ \cos(2\theta)\end{pmatrix}$$
$$= \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}\begin{pmatrix}c_1 & s_1\end{pmatrix}\begin{pmatrix}-c_1s_2 - c_2s_1\\ c_1c_2 - s_1s_2\end{pmatrix}$$
$$= -s_2\sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}.$$

Note that there is no need to compute explicitly θ, θ_1 or θ_2. We now briefly describe the underlying diagonal uniformization algorithm. The mean of Σ_V diagonal coefficients being equal to τ, these indices sets are not empty: $iInf \stackrel{def}{=} \{\ l \in \{1,\ ...,\ c\} \mid \Sigma_{V,l,l} < \tau\}$ and $iSup \stackrel{def}{=} \{\ l \in \{1,\ ...,\ c\} \mid \Sigma_{V,l,l} > \tau\}$. Taking one index j from $iInf$ and the other one i from $iSup$ guarantees the condition of Th. 3.1, which allows to set $\Sigma_{V,j,j}$ to the value τ. The index j can then be removed from $iInf$ and as $\Sigma_{V,i,i}$ is set to $a + d - \tau$, the index i reassigned to $iInf$ if $\tau > \frac{a+d}{2}$ or in $iSup$ if $\tau < \frac{a+d}{2}$. The number of diagonal coefficients of Σ_V different from τ has been decreased by one. Finally, the necessary number of iterations to completely empty $iInf$ and $iSup$, i.e. uniformizing Σ_V diagonal, is bounded by $c-1$. The method is summarized in Algorithm 1 where $\mathrm{pop}(list)$ and $\mathrm{add}(list, e)$ are subroutines to delete and return the first element of $list$, resp. to add e in $list$.

[3]Th. 3.1 uses only a sufficient condition. A weaker necessary and sufficient one to guarantee $|c_2| \leq 1$ and $s_2 \in [0,1]$ is $\frac{a+d}{2} - \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2} \leq \tau \leq \frac{a+d}{2} + \sqrt{\left(\frac{a-d}{2}\right)^2 + b^2}$.

Algorithm 1 Diagonal Uniformization algorithm (UnifDiag)

Inputs : Σ_V ($c \times c$, symmetric), tolerance: tol
$R \leftarrow I_c$ // $c \times c$ Identity matrix; $\tau \leftarrow \mathrm{Tr}(\Sigma_V)/c$; $it = 0$
$iInf = \{\ l \in \{1,\ ...,\ c\} \mid \Sigma_{V,l,l} < \tau - tol\}$
$iSup = \{\ l \in \{1,\ ...,\ c\} \mid \Sigma_{V,l,l} > \tau + tol\}$
while $it < c-1$ & not isEmpty($iInf$) & not isEmpty($iSup$) **do**
  // Givens rotation parameters computation:
  $j \leftarrow pop(iInf)$; $i \leftarrow \mathrm{pop}(iSup)$; $a \leftarrow \Sigma_V[j,j]$; $b \leftarrow \Sigma_V[i,j]$; $d \leftarrow \Sigma_V[i,i]$; c, s (Th. 3.1); $it \leftarrow it + 1$
  // Σ_V update:
  $row_j \leftarrow \Sigma_V[j,:]$; $row_i \leftarrow \Sigma_V[i,:]$
  $\Sigma_V[j,:] = c \times row_j - s \times row_i$;
  $\Sigma_V[i,:] = s \times row_j + c \times row_i$
  $\Sigma_V[:,j] = \Sigma_V[j,:]$; $\Sigma_V[:,i] = \Sigma_V[i,:]$
  $\Sigma_V[j,j] = a'$; $\Sigma_V[i,i] = d'$; $\Sigma_V[j,i] = b'$ (Th. 3.1)
  // Rotation update:
  $col_j \leftarrow R[:,j]$; $col_i \leftarrow R[:,i]$
  $R[:,j] = c \times col_j - s \times col_i$
  $R[:,i] = s \times col_j + c \times col_i$
  // Indices list update:
  if $\frac{a+d}{2} < \tau - tol$ **then**
    $\mathrm{add}(iInf, i)$
  if $\frac{a+d}{2} > \tau + tol$ **then**
    $\mathrm{add}(iSup, i)$
return R

4. COMPLEXITY ANALYSIS OF EXISTING WORKS

Our algorithm requires the storage of two $c \times c$ matrices besides W_t and R_t obviously: one with OPAST to obtain W_t and $\Sigma_{V,t}$ for R_t. One update with OPAST for W_t and $\Sigma_{V,t}$ costs $4dc + O(c^2)$. Then, to compute R_t, at most $c-1$ Givens rotations are needed, each implying four column or row multiplications i.e. $4c$ flops. So the final time complexity of our algorithm is $4dc + O(c^2)$.

We compare here the spatial and time costs of our method with, to the best of our knowledge, the only online unsupervised method: **Online Sketching Hashing** (OSH) [11] which

is 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. Without counting the projection matrix and the rotation, OSH needs spatially to maintain the sketch S which costs $O(d \times l)$ with $c \ll l \ll d$. The SVD decomposition then needs $O(dl + l^2)$ space. In comparison, we only need $O(c^2)$. For each round, OSH takes $O(dl^2 + l^3)$ time to learn the principal components, i.e. $O(dl + l^2)$ for each new data seen. We also compare with **IsoHash** [8]. Although it counts as an offline method because no technique is proposed to approximatively estimate the principal subspace, IsoHash rotation can be applied after for instance OPAST. IsoHash rotation computation involves an integration of a differential equation using Adams-Bashforth-Moulton PECE solver which costs $O(c^3)$ time. Even if c is small in comparison to d and the complexities do not either depend on n, our model has the advantages to have a lower time cost and to be much more simple than IsoHash. Thus, our method shows advantages in terms of spatial and time complexities over OSH and IsoHash. 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

Experiments are made on CIFAR-10 (`http://www.cs.toronto.edu/~kriz/cifar.html`) and GIST1M sets (`http://corpus-texmex.irisa.fr/`). CIFAR-10 contains 60000 32×32 color images equally divided into 10 classes. 960-D GIST descriptors were extracted from data. GIST1M contains 1 million 960-D GIST descriptors, from which 60000 instances were randomly chosen from the first half of the learning set. Quality of hashing has been assessed on the nearest neighbor (NN) search task performed on the binary codes instead of the initial descriptors. 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. 1000 queries were randomly sampled and the remaining data are used as training set. We compared our method to three online baseline methods that follow the basic hashing scheme $\Phi(x_t) = \text{sgn}(\tilde{W}_t x_t)$, where the projection matrix $\tilde{W}_t \in \mathbb{R}^{c\times d}$ is determined according to the chosen method: 1) **OSH** 2) **RandRot-OPAST**: W_t is the PCA matrix obtained with OPAST and R_t a constant random rotation. 3) **IsoHash-OPAST**: R_t is obtained with IsoHash. 4) **UnifDiag-OPAST**: R_t is obtained with UnifDiag. For OSH, the number of chunks is set to 200 and $l = 50$. Fig. 1 and 2 (best viewed in color) show the Mean Average Precision (mAP) [29] for both datasets for $c = 32$ (similar results are obtained for $c \in [8, 16, 64]$) averaged over 5 random training/test partitions. Our algorithm outperforms all the compared methods. Code on GitHub: `annemorvan/UnifDiagStreamBinSketching/`.

6. CONCLUSION

We introduced a novel method for learning distance-preserving binary embeddings of high-dimensional data streams with convergence guarantees. Unlike classical state-of-the-art methods, our algorithm does not need to store the whole dataset and enables to obtain without delay a binary code as a new data point is seen. Our approach shows promising results as evidenced by the experiments. It can achieve better accuracy than state-of-the-art online unsupervised methods while saving considerable computation time and spatial requirements. Besides, the Givens rotations, that are a classical tool for QR factorization, singular and eigendecomposition or joint diagonalization, can also be used for uniformizing the diagonal of a symmetric matrix via an original Givens angle tuning technique. 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 is 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 while preserving the accuracy.

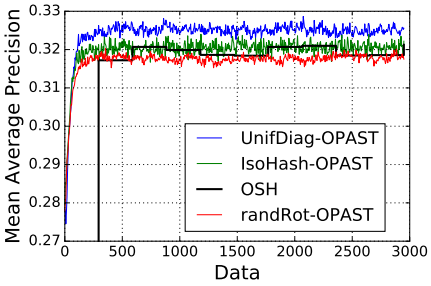

Fig. 1. mAP for $c = 32$ on CIFAR-10.

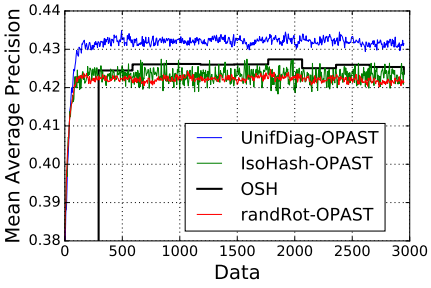

Fig. 2. mAP for $c = 32$ on GIST-1M.

7. REFERENCES

[1] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Commun. ACM*, , no. 1, pp. 117–122, 2008.

[2] K. Terasawa and Y. Tanaka, "Spherical LSH for approximate nearest neighbor search on unit hypersphere," in *WADS*, 2007, pp. 27–38.

[3] A. Andoni, P. Indyk, I. Laarhoven, T.and Razenshteyn, and L. Schmidt, "Practical and optimal LSH for angular distance," in *NIPS*, 2015, pp. 1225–1233.

[4] 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*, 2017, pp. 1020–1029.

[5] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing.," in *NIPS*, 2008, pp. 1753–1760.

[6] J. Wang, S. Kumar, and S. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, , no. 12, pp. 2393–2406, 2012.

[7] 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*, , no. 12, pp. 2916–2929, 2013.

[8] W. Kong and W. Li, "Isotropic hashing," in *NIPS*, pp. 1646–1654. 2012.

[9] F. Yu, S. Kumar, Y. Gong, and S. Chang, "Circulant binary embedding," in *ICML*, 2014.

[10] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010, pp. 3304–3311.

[11] C. Leng, J. Wu, J.and Cheng, X. Bai, and H. Lu, "Online sketching hashing.," in *CVPR*, 2015, pp. 2503–2511.

[12] J. Wang, W. Liu, S. Kumar, and S. Chang, "Learning to hash for indexing big data - a survey," *Proceedings of the IEEE*, , no. 1, pp. 34–57, 2016.

[13] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *NIPS*, pp. 1509–1517. 2009.

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

[15] W. Liu, J. Wang, and S. Chang, "Hashing with graphs," in *ICML*, 2011.

[16] Y. Lee, "Spherical hashing," in *CVPR*, 2012, pp. 2957–2964.

[17] W. Liu, C. Mu, S. Kumar, and S. Chang, "Discrete graph hashing," in *NIPS*, pp. 3419–3427. 2014.

[18] R. Raziperchikolaei and M. Á. Carreira-Perpiñán, "Optimizing affinity-based binary hashing using auxiliary coordinates," in *NIPS*, 2016, pp. 640–648.

[19] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang, "Supervised hashing with kernels," in *CVPR*, 2012, pp. 2074–2081.

[20] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *CVPR*, 2015, pp. 3270–3278.

[21] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *ICML*, 2015, pp. 2285–2294.

[22] C. Leng, J. Wu, J. Cheng, X. Zhang, and H. Lu, "Hashing for distributed data," in *ICML*, 2015, pp. 1642–1650.

[23] L. Huang, Q. Yang, and W. Zheng, "Online hashing," in *IJCAI*, 2013, pp. 1422–1428.

[24] F. Çakir, K. He, S.A. Bargal, and S. Sclaroff, "MIHash: Online hashing with mutual information," in *ICCV*, Oct 2017.

[25] K. Abed-Meraim, A. Chkeif, and Y. Hua, "Fast orthonormal past algorithm," *IEEE Signal Processing Letters*, , no. 3, pp. 60 – 62, 2000.

[26] J. Feng, H. Xu, and S. Yan, "Online robust pca via stochastic optimization.," in *NIPS*, 2013, pp. 404–412.

[27] W. Yang and H. Xu, "Streaming sparse principal component analysis.," in *ICML*, 2015, pp. 494–503.

[28] G. H. Golub and H. A. van der Vorst, "Eigenvalue computation in the 20th century," *Journal of Computational and Applied Mathematics*, , no. 12, pp. 35 – 65, 2000, Numerical Analysis 2000. Vol. III: Linear Algebra.

[29] C.D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.