Online Sketching Hashing

Cong Leng[1], Jiaxiang Wu[1], Jian Cheng[1], Xiao Bai[2], Hanqing Lu[1]
[1]Institute of Automation, Chinese Academy of Sciences. [2]School of Computer Science and Engineering, Beihang University.

Recently, hashing based approximate nearest neighbor (ANN) search has attracted much attention. Extensive new algorithms have been developed and successfully applied to different applications. However, two critical problems are rarely mentioned. First, in many real-world applications, data becomes available continuously in streaming fashion. However, most of the existing hashing models are based on a batch learning fashion. That is to say, when new data arrives, they have to accumulate all the available data and re-train new hash functions, which is apparently a less efficient learning manner for streaming data. Second, for truly large scale datasets, data is usually stored on a distributed disk group and is too large to be read into memory. Moreover, one processor is often incapable of handling the large scale datasets in a feasible amount of time.

In this paper, we propose a novel online hashing approach to address the two problems mentioned above simultaneously. The proposed method is largely inspired by the idea of data sketching [4, 5]. We assume that the data is available in a stream form. Let $\mathcal{D}_i$ denote the data chunk received at round i, where $i = \{1, 2, \cdots\}$. In particular, $\mathcal{D}_i = [x_1^i, x_2^i, \cdots, x_{m_i}^i] \in \mathbb{R}^{d\times m_i}$ contains m_i samples. The mean of the data chunk $\mathcal{D}_i$ is denoted by $\overline{\mathcal{D}_i}$. X_t denotes the data matrix accumulated from round 1 to round t. μ_t is the mean of data in X_t and n_t is the data size. Our approach maintains a small size sketch for the streaming data online, and we demonstrate that the hash functions can be efficiently learned based on this sketch.

Suppose we have a $d \times n$ matrix $X = [x_1, x_2, \cdots, x_n]$ where each column x_i is a sample in the dataset. Denote $W = [w_1, w_2, \cdots, w_r] \in \mathbb{R}^{d\times r}$, then the objective of PCA hashing [1, 3, 7] can be formally written as:

$$\max_{W\in\mathbb{R}^{d\times r}} \quad tr(W^T(X-\mu)(X-\mu)^T W) \qquad \text{s.t.} \quad W^T W = I_r \tag{1}$$

where $\mu = \frac{1}{n}\sum_{j=1}^{n} x_j$ is the mean of all the data. The notation $(X-\mu)$ means the matrix $[x_1-\mu, x_2-\mu, \cdots, x_n-\mu]$, which is equivalent to centering the data. However, this is obviously a batch based learning method and suffers from the two limitations we mentioned above.

A sketch of matrix P is another matrix Q which is much smaller than P, but still preserves the properties of interest. In this way, the storage of the sketch Q will be much easier, and the computations can be performed much faster than with the original P [4, 5]. Given a matrix $P \in \mathbb{R}^{d\times n}$, we aim to maintain a much smaller matrix $Q \in \mathbb{R}^{d\times l}$ with $l \ll n$ as an approximation to P. The goal is to track an ε-approximation to the norm of matrix along any direction, i.e., $|\|P^T x\|^2 - \|Q^T x\|^2| \le \varepsilon \|P\|_F^2, \quad \forall x, \ \|x\| = 1$. The latest significant effort is represented by Frequent Directions (FD) proposed by Liberty [5]. Inspired by the works in finding frequent items, Liberty investigated how to apply the Misra-Gries technique [6] to matrix sketching. FD provides a tight bounds for its performance. Formally, we have

$$0 \le \|PP^T - QQ^T\|_2 \le \frac{2}{l}\|P\|_F^2 \tag{2}$$

We attempt to employ the favorable property that $PP^T \approx QQ^T$ in [5] to handle the scalability and streaming data issue in hashing. A very straightforward way is sketching the matrix $X-\mu$ in Eq.(5) with [5] so that we can get a significantly smaller sketch Y which approximates $X-\mu$ well with $YY^T \approx (X-\mu)(X-\mu)^T$. However, it is infeasible because the data is continuously changing, and therefore the mean of data μ changes too. When a new data chunk $\mathcal{D}_t$ arrives at round t, since the mean of the dataset changes to μ_t, we need to re-sketch all the data $[\mathcal{D}_1 - \mu_t, \mathcal{D}_2 - \mu_t, \cdots, \mathcal{D}_t - \mu_t]$. In order to avoid the mean-varying problem, one can augment every data chunk with a virtual sample, which is carefully chosen to correct the time-varying mean. Specifically, for a stream X_t we design a matrix E_t as $E_t = [\widehat{\mathcal{D}_1}, \widehat{\mathcal{D}_2}, \cdots, \widehat{\mathcal{D}_t}]$ where

$$\widehat{\mathcal{D}_1} = \mathcal{D}_1 - \overline{\mathcal{D}_1},$$
$$\widehat{\mathcal{D}_i} = [\mathcal{D}_i - \overline{\mathcal{D}_i}, \sqrt{\frac{n_{i-1}m_i}{n_{i-1}+m_i}}(\overline{\mathcal{D}_i} - \mu_{i-1})] \tag{3}$$

In this way, at any round t, we have

$$E_t E_t^T = cov(X_t) \tag{4}$$

Thus, it allows us to find a sketch Y_t which approximates E_t well and then learn hash functions online based on this sketch. The overall accumulated time complexity is $O(n_t dl + tdl^2 + tl^3)$, and the overall space complexity is $O(ld + l^2 + dr)$.

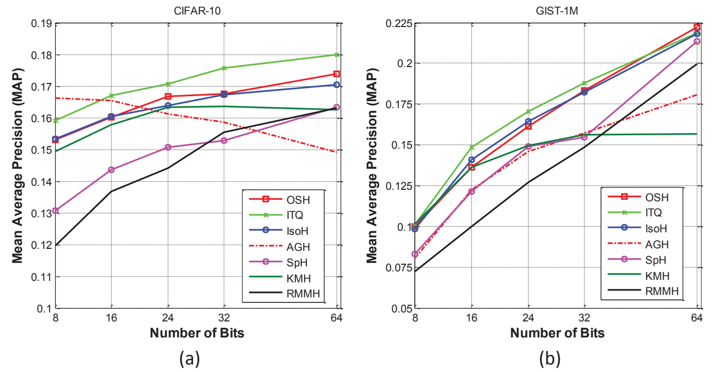

Figure 1: Hamming ranking performance: mean average precision (MAP) of different algorithms with different code lengthes on (a) CIFAR-10 and (b) GIST-1M. (Best viewed in color)

We test the proposed methods on two benchmarks CIFAR-10 and GIST-1M. Fig.1 reports the most interesting results we think. In this experiment, we find a sketch of size 200 for the training data and then learn hash functions on this sketch. We find that our method achieve comparable performance to ITQ and outperforms most of state-of-the-arts hash methods. It implies that the sketch (even of size 200) can preserve sufficient information for hash functions learning. We also evaluate the training time of our method. By dividing the data into 100 chunks, the proposed method achieves about three times speed-up on both datasets than OKH [2], which verifies the efficiency of our method in online setting.

[1] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI*, 2013.

[2] Long-Kai Huang, Qiang Yang, and Wei-Shi Zheng. Online hashing. In *IJCAI*. AAAI Press, 2013.

[3] Weihao Kong and WuJun Li. Isotropic hashing. In *NIPS*, 2012.

[4] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *STOC*. ACM, 2009.

[5] Edo Liberty. Simple and deterministic matrix sketching. In *SIGKDD*. ACM, 2013.

[6] Jayadev Misra and David Gries. Finding repeated elements. *Science of computer programming*, 2(2):143–152, 1982.

[7] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.