Near Optimal Frequent Directions for Sketching Dense and Sparse Matrices

Zengfeng Huang [1]

Abstract

Given a large matrix $A \in \mathbb{R}^{n\times d}$, we consider the problem of computing a sketch matrix $B \in \mathbb{R}^{\ell\times d}$ which is significantly smaller than but still well approximates A. We are interested in minimizing the *covariance error* $\|A^TA - B^TB\|_2$. We consider the problems in the streaming model, where the algorithm can only make one pass over the input with limited working space. The popular Frequent Directions algorithm of (Liberty, 2013) and its variants achieve optimal space-error tradeoff. However, whether the running time can be improved remains an unanswered question. In this paper, we almost settle the time complexity of this problem. In particular, we provide new space-optimal algorithms with faster running times. Moreover, we also show that the running times of our algorithms are near-optimal unless the state-of-the-art running time of matrix multiplication can be improved significantly.

1. Introduction

For large-scale matrix computations, exact algorithms are often too slow, so a large body of works focus on designing fast randomized approximation algorithms. To speedup the computation, matrix sketching is a commonly used technique, e.g. (Sarlos, 2006; Clarkson & Woodruff, 2013; Avron et al., 2013; Chierichetti et al., 2017). In real-world applications, the data often arrives in a streaming fashion and it is often impractical or impossible to store the entire data set in the main memory.

In this paper, we study online streaming algorithms for maintaining matrix sketches with small *covariance errors*. In the *streaming model*, the rows of the input matrix arrive one at a time; the algorithm is only allowed to make one pass over the stream with severely limited working space, which is required to maintain a sketch continuously. This problem has received lots of attention recently (Liberty, 2013; Ghashami & Phillips, 2014; Woodruff, 2014; Ghashami et al., 2016; Wei et al., 2016).

The popular *Frequent Directions* algorithms (Liberty, 2013; Ghashami et al., 2016) achieve optimal tradeoff between space usage and approximation error (Woodruff, 2014), which have found lots of applications in online learning, e.g., (Boutsidis et al., 2015; Karnin & Liberty, 2015; Leng et al., 2015; Huang & Kasiviswanathan, 2015; Luo et al., 2016; Calandriello et al., 2017), and other problems (Song et al., 2015; Ye et al., 2016; Kim et al., 2016). However, it is unclear whether their running times can be improved; one might hope to get linear (in sparsity) time algorithms, which is possible for many matrix problems, e.g. (Clarkson & Woodruff, 2013). This paper is motivated by the following question:

> *Is there an input sparsity time Frequent Directions, which achieves the same optimal space-error tradeoff?*

1.1. Problem definitions

Given a matrix $A \in \mathbb{R}^{n\times d}$, we want to compute a much smaller matrix $B \in \mathbb{R}^{\ell\times d}$, which has low *covariance error*, i.e., $\|A^TA - B^TB\|_2$.

Definition 1 (Covariance Sketch)**.** *For any* $0 < \varepsilon < 1$*, and integer* $0 \le k \le \mathrm{rank}(A)$*, we will call* B *an* (ε, k)-cov-sketch *of* A*, if the covariance error*[1]

$$\|A^TA - B^TB\|_2 \le \varepsilon\|A - [A]_k\|_F^2. \tag{1}$$

Here $\|\cdot\|_2$ and $\|\cdot\|_F$ are the *spectral norm* and *Frobenius norm* of matrices; $[A]_k$ is the best rank-k approximation to A. We will use $\pi_B^k(A)$ to denote the projection of A on the top-k singular vectors of B, i.e. $\pi_B^k(A) = AVV^T$, where the columns of V are the top-k right singular vectors of B.

Definition 2 (Projection error)**.** *The projection error of* B *with respect to* A *is defined as* $\|A - \pi_B^k(A)\|_F^2$*.*

Note $\pi_B^k(A)$ is a rank-k matrix, thereby the projection error is at least $\|A - [A]_k\|_F^2$. It is proved in (Ghashami & Phillips, 2014) that one can obtain relative projection error from small covariance error. We include a proof of the next lemma in the supplementary material.

[1]School of Data Science, Fudan University, China. Correspondence to: Zengfeng Huang <huangzf@fudan.edu.cn>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

[1]for $k = 0$, we define $[A]_0 = \mathbf{0}$

Lemma 1 (covariance error to projection error (Ghashami & Phillips, 2014) (modified))**.**

$$kA \quad {}^{k}_{B}(A)k_F^2 \quad kA \quad [A]_k k_F^2 + 2k \quad kA^T A \quad B^T B k_2:$$

Therefore, any $(\frac{\varepsilon}{2k}; k)$-cov-sketch B has projection error

$$kA \quad {}^{k}_{B}(A)k_F^2 \quad (1 + \varepsilon)kA \quad [A]_k k_F^2: \qquad (2)$$

We will often refer to such sketches as $(\varepsilon; k)$-proj-sketches.

Modern data matrices are often large and sparse. So we will always assume n and d are very large, typically $d \quad n$, and $\mathrm{nnz}(A) \quad nd$, where $\mathrm{nnz}(A)$ in the number of nonzero entries in A. Moreover, we assume that each entry of A is representable by $O(\log(nd))$ bits. To simplify the analysis, we assume the entries of A are integers of magnitude at most $\mathrm{poly}(nd)$; the general case can be reduced to this, see e.g. (Boutsidis et al., 2016).

1.2. Previous results

In the row-wise update streaming model, Liberty's *Frequent Direction* (FD) algorithm (Liberty, 2013), with an improved analysis in (Ghashami & Phillips, 2014), maintains an $(\ ; k)$-cov-sketch $B \ 2 \ \mathbb{R}^{\ell \times d}$ at any time, where $\ell = O(k + \ ^{-1})$. The algorithm uses $O(d\ell)$ space and runs in $O(nd\ell)$ time. For sparse matrices, the running time of FD is improved to $O(\mathrm{nnz}(A)\ell \log d + \mathrm{nnz}(A) \log n + n\ell^2)$ by Ghashami et al. (Ghashami et al., 2016). Set $= \varepsilon/2k$ (or $\ell = O(k/\varepsilon)$), and by Lemma 1, B is a $(\varepsilon; k)$-proj-sketch. Now, B contains $O(k/\varepsilon)$ rows, the space and the running time become $O(dk/\varepsilon)$ and $O(\mathrm{nnz}(A)k\varepsilon^{-1} \log d + \mathrm{nnz}(A) \log n + nk^2\varepsilon^{-2})$ respectively. It was shown by Woodruff (Woodruff, 2014) that the space used by FD is optimal for both covariance error and projection error. A natural question is if the running time can be improved. In particular,

> *Is there an input sparsity time algorithm, i.e., in time* $O(\mathrm{nnz}(A) + (n + d) \ \mathrm{poly}(k \ ^{-1}))$*, which achieves the same guarantee as* FD?

1.3. Our contributions

This paper almost settles the above question. Our main contributions are summarized as follows.

1. We show that $o(\mathrm{nnz}(A)k)$ time is likely very difficult to achieve, as it will imply a breakthrough in fast matrix multiplication. In particular, we prove that computing an $(O(1); k)$-cov-sketch $B \ 2 \ \mathbb{R}^{O(k)\times d}$ of A is as hard as left multiplying A by an arbitrary matrix $C \ 2 \ \mathbb{R}^{k\times n}$.

2. We give a new space-optimal streaming algorithm with $O(ndk) + O(d \ ^{-3})$ running time to compute $(\ ; k)$-cov-sketches for dense matrices, which improves the original FD algorithm for small . The running time is optimal up to lower order terms, provided matrix multiplication cannot be improved significantly.

3. We then give a new space-optimal streaming algorithm with $O(\mathrm{nnz}(A)k + \mathrm{nnz}(A) \log n) + O(nk^3 + d \ ^{-3})$ running time to compute $(\ ; k)$-cov-sketches for sparse matrices. We separate the dependence of $1/$ from $\mathrm{nnz}(A)$, which improves the results of (Ghashami et al., 2016) for small . In particular, computing an $(\varepsilon; k)$-proj-sketch, our algorithm only needs $O(\mathrm{nnz}(A)k)$ time (ignoring lower order terms) as opposed to $O(\mathrm{nnz}(A)k\varepsilon^{-1} \log d)$ in (Ghashami et al., 2016) (see Table 1). Even when is small, our algorithm improves a $\log d$ factor. Moreover, for $k = (\log n)$, the running time of our algorithm matches the lower bound.

1.4. Other related works

The problem of computing $(\ ; k)$-cov-sketches was also studied in the sliding window streaming model (Wei et al., 2016) and distributed models (Ghashami et al., 2014; Huang et al., 2017). A closely related problem, namely *approximate PCA*, was studied in (Kannan et al., 2014; Liang et al., 2014; Boutsidis et al., 2016; Zhang et al., 2015). (Clarkson & Woodruff, 2009) studied other streaming numerical linear algebra problems.

1.5. Matrix preliminaries and notations

We always use n for the number rows, and d for the dimension of each row. For a d-dimensional vector x, $\|x\|$ is the ℓ_2 norm of x. We use x_i to denote the ith entry of x, and $\mathrm{Diag}(x) \ 2 \ \mathbb{R}^{d\times d}$ is a diagonal matrix such that the ith diagonal entry is x_i. Let $A \ 2 \ \mathbb{R}^{n\times d}$ with $n > d$, we use A_i to denote the ith row of A, and $a_{i,j}$ for the $(i; j)$-th entry of A. $\mathrm{nnz}(A)$ is the number of non-zero entries in A, and $\mathrm{rows}(A)$ is the number of rows in A. We write the (reduced) singular value decomposition of A as $(U; \ ; V) = \mathrm{SVD}(A)$. The computation time of standard SVD algorithms is $O(nd^2)$. We use $\|A\|_2$ or $\|A\|$ to denote the spectral norm of A, which is the largest singular value of A, and $\|A\|_F$ for the *Frobenius Norm*, which is $\sqrt{\sum_{i,j} a_{i,j}^2}$. For $k \quad \mathrm{rank}(A)$, we use $[A]_k$ to denote the best rank k approximation of A. We define $[A]_0 = \mathbf{0}$. $[A; B]$ is the matrix formed by concatenating the rows of A and B. We use $O()$ to hide $\mathrm{polylog}(ndk)$ factors.

1.6. Tools

Frequent Directions. We will use the Frequent Directions (FD) algorithm by Liberty (Liberty, 2013), denoted as $\mathrm{FD}(A; \ ; k)$, and the main result is summarized in the following theorem.

Theorem 1 ((Liberty, 2013))**.** *Given* $A \ 2 \ \mathbb{R}^{n\times d}$*, in one pass,* $\mathrm{FD}(A; \ ; k)$ *processes* A *in* $O(nd(k + \ ^{-1}))$ *time*

	Time	
	(α, k)-cov	(ε, k)-proj
Lib13, GP13 (Liberty, 2013; Ghashami & Phillips, 2014)	$O(ndk + nd/\alpha)$	$O(ndk/\varepsilon)$
New	$O(ndk)$	$O(ndk)$
GLP16 (Ghashami et al., 2016)	$O(\text{nnz}(A)k \log d + \text{nnz}(A) \log d/\alpha)$	$O(\text{nnz}(A)k \log d/\varepsilon)$
New	$O(\text{nnz}(A)k + \text{nnz}(A) \log n)$	$O(\text{nnz}(A)k + \text{nnz}(A) \log n)$

Table 1. Running time for streaming (α, k)-cov-sketch and (ε, k)-proj-sketch algorithms. The low order terms are omitted.

and $O(d(k + \quad ^{-1}))$ space. It maintains a matrix B 2 $\mathbb{R}^{O(k+\alpha \quad ^{1})\times d}$ such that $kA^TA \quad B^TBk_2 \qquad kA \quad [A]_k k_F^2$:

Row sampling. We provide a result about row sampling, which is analogous to a result from (Drineas et al., 2006). The difference is that they use *iid sampling*, i.e., each row of B is an iid sample from the rows of A. On the other hand, we use *Bernoulli sampling*, i.e., sample each A_i independently with some probability q_i, and B is the set of sampled rows. *Bernoulli sampling* can easily be combined with FD in the streaming model. The proof is essentially the same as that for iid sampling, which can be found in the supplementary material.

Theorem 2. *For any A 2 $\mathbb{R}^{n\times d}$ and F > 0, we sample each row A_i with probability $p_i \quad \frac{\|A_i\|^2}{\alpha^2 F}$; if it is sampled, scale it by $1=\sqrt{p_i}$. Let B be the (rescaled) sampled rows, then w.p. 0:99, $kA^TA \quad B^TBk_2 \quad 10 \quad \sqrt{F}kAk_F$, and $kBk_F \quad 10kAk_F$. The expected number of rows sampled is $O(\frac{\|A\|_F^2}{\alpha^2 F})$.*

Input-sparsity time lower rank approximation algorithm. There are $\text{nnz}(A)$ (omitting lower order terms) time algorithms (Clarkson & Woodruff, 2013), which output a matrix Z consists of k orthonormal rows such that $kA \quad Z^TZAk_F^2 \quad (1 + \text{"})kA \quad [A]_k k_F^2$: For our application, we only need a constant approximation, and only require Z to contain $O(k)$ rows. For this purpose, we give a simplified algorithm with slightly better running time than directly applying the results from (Clarkson & Woodruff, 2013). We require high success probability, which can be achieved using similar techniques as in (Boutsidis et al., 2016). The proof of the following theorem can be found in the supplementary file.

Theorem 3 (weak low rank approximation)**.** *For any integers $\grave{}; d$, given A 2 $\mathbb{R}^{\ell\times d}$, there is an algorithm that uses $O(\text{nnz}(A) \log(1= \text{"}))$ + $O(\grave{}k^3)$ time and $O(\grave{}(k^2 + \log \frac{1}{\delta}))$ space, and outputs a matrix Z 2 $\mathbb{R}^{O(k)\times \ell}$ with orthonormal rows such that with probability 1 , $kA \quad Z^TZAk_F^2$ $O(1)kA \quad [A]_k k_F^2$:*

2. Time lower bound

In this section, we provide a conditional lower bound for our problem based on the idea of (Musco & Woodruff, 2017). We prove that the existence of algorithms which compute an $(O(1); k)$-cov-sketch in time $o(\text{nnz}(A)k)$ implies a breakthrough in matrix multiplication, which is likely very difficult. In fact, the lower bound holds even for offline algorithms without constraints on working space.

Theorem 4. *Assume there is an algorithm A , which, given any A 2 $\mathbb{R}^{n\times d}$ with polynomially bounded integer entries, returns B 2 $\mathbb{R}^{O(k)\times d}$ in time $o(\text{nnz}(A)k)$ such that*

$$kA^TA \quad B^TBk_2 \qquad kA \quad [A]_k k_F^2;$$

for some constant error parameter . Then there is an $o(\text{nnz}(M)k) + O(dk^2)$ time algorithm for multiplying arbitrary polynomially bounded integer matrices M^T 2 $\mathbb{R}^{(d-k)\times n}$, C 2 $\mathbb{R}^{n\times k}$.

Proof. For any matrices M 2 $\mathbb{R}^{n\times(d-k)}$ and C 2 $\mathbb{R}^{n\times k}$ with integer entries in [$U; U$], let A 2 $\mathbb{R}^{n\times d}$ be the matrix which is a concatenation of the columns of M and wC, i.e., $A = [M; wC]$: Here w is large number will be determined later. We have $kA \quad [A]_k k_F^2 \quad kMk_F^2 \quad ndU^2$ and

$$A^TA = \begin{bmatrix} M^TM & wM^TC \\ wC^TM & w^2C^TC \end{bmatrix}:$$

We assume A is an algorithm with running time T, which can compute a sketch matrix B 2 $\mathbb{R}^{O(k)\times d}$ of A such that

$$kA^TA \quad B^TBk_2 \qquad kA \quad [A]_k k_F^2 \qquad U^2nd;$$

for some constant error parameter .

The spectral norm of a matrix N is the largest singular value, which can be equivalently defined as $kNk_2 = \max_{x,y:\|x\|=\|y\|=1} x^TNy$; thereby $N_{i,j} = e_i^TNe_j \quad kNk_2$ for all $i; j$. It follows that $(A^TA \quad B^TB)_{i,j} \qquad U^2nd$, meaning the corresponding block of B^TB is an entry-wise approximation to wM^TC within additive error U^2nd.

Now if w is a large integer, say $w = d3 \quad U^2nde$, we can recover M^TC from B^TB exactly by rounding the numbers in B^TB to their nearest integers (as M^TC is an integer matrix). Note B^TB can be computed in time $O(dk^2)$ given B and the rounding can be done in $O(dk)$, so using A, the exact integer matrix multiplication M^TC can be computed in time $O(T + dk^2)$. Therefore, we have proved that if $T = o(\text{nnz}(A)k) = o(\text{nnz}(M)k + dk^2)$, then M^TC can

be computed in time $o(\mathrm{nnz}(M)k) + O(dk^2)$, which will be a breakthrough in fast matrix multiplication. We remark that all the integers in our reduction are at most $\mathrm{poly}(nd)$ in magnitude, as long as $U = \mathrm{poly}(nd)$, so our reduction works for any $M; C$ with polynomially bounded entries. □

3. Algorithm for dense matrices

Theorem 5 (FFDdense($A;$ $;k$)). *Given a matrix A 2 $\mathbb{R}^{n\times d}$, $0 <$ < 1 and 0 k d, FFDdense($A;$ $;k$) processes A in one pass using $O(ndk) + O(d^{-3})$ time and $O(dk + d^{-1}))$ space, and maintains a matrix B. With probability 0:99, it holds that kA^TA B^TBk_2 kA $[A]_k k_F^2$.*

Overview of the algorithm. To speed up FD, we will use the idea of adaptive random sampling. Let us first review the standard FD algorithm. Given an integer parameter ℓ d, the algorithm always maintains a matrix B with at most 2ℓ rows at any time. When a new row v arrives, it processes the row using FDShrink($B; v; \ell$) (Algorithm 3). In this procedure, we first append a after B; if B has no more than 2ℓ rows we do nothing, and otherwise we do a DenseShrink operation (Algorithm 1) on B, which halves the number of rows in B (after removing zero rows). It was proved in (Liberty, 2013) and (Ghashami & Phillips, 2014) that for $\ell = k + {}^{-1}$, we have

$$kA^TA \quad B^TBk_2 \quad kA \quad [A]_k k_F^2.$$

Since each SVD computation in DenseShrink takes $O(d\ell^2)$ time, and there are totally n/ℓ SVD computations (SVD is applied every ℓ rows), the running time is $O(nd\ell) = O(nd(k + {}^{-1}))$. Our goal is to separate nd from ${}^{-1}$ in the running time.

Algorithm 1 DenseShrink

Input: B 2 $\mathbb{R}^{2\ell\times d}$.

1: Compute $[U;$ $;V] = \mathrm{SVD}(B)$, and $=$ $_{\ell,\ell}$.
2: $\hat{}$ $= \sqrt{\max(\ ^2 \quad ^2 I_\ell; 0)}$ ▶ $\mathrm{ReLu}(x) = \max(x; 0)$
3: **return** $B = \hat{}V^T$

Algorithm 2 DenseShrinkR

Input: B 2 $\mathbb{R}^{2\ell\times d}$.

1: Compute $[U;$ $;V] = \mathrm{SVD}(B)$, and $=$ $_{\ell,\ell}$.
2: $\hat{}$ $= \sqrt{\max(\ ^2 \quad ^2 I_\ell; 0)}$
3: $= \sqrt{\ ^2 \quad \hat{}^2}$ ▶ $^2 = \ ^2 + \hat{}^2$
4: **return** $B = \hat{}V^T$, and V^T

To achieve this, we first compute a coarse approximation using FD by invoking $B = \mathrm{FD}(A; k; \frac{1}{2k})$, which takes $O(ndk)$ time. The key idea here is that in each DenseShrink operation, after shrinking B, we also return the residual; we call this modified shrinking operation DenseShrinkR (see Algorithm 2). Let C be the matrix which is the concatenation of all residuals return from DenseShrinkR. We will show $A^TA = B^TB + C^TC$ and kCk_F^2 kA $[A]_k k_F^2$. We then refine the answer by computing an approximation to C. Since the norm of C is small, random sampling suffices. To save space, the sampled rows will be fed to a standard FD algorithm. See Algorithm 4 for detailed description of the algorithm.

Algorithm 3 FDShrink

Input: B 2 $\mathbb{R}^{\ell'\times d}$, $\mathbf{v}$ 2 $\mathbb{R}^d$, and an integer ℓ
▶ it always holds that $\ell' < 2\ell$.

1: $B = [B; \mathbf{v}]$
2: **If** $\ell' + 1 = 2\ell$, **then** $B = \mathrm{DenseShrink}(B; \ell)$.
3: **return** B

Algorithm 4 FFDdense

Input: A 2 $\mathbb{R}^{n\times d}$, $0 <$ < 1, and integer k d.

1: $F = 0$, $\ell = 3k$, $Q =$ **empty**
2: **for** $i = 1$ **to** n **do**
3: Append A_i after B
4: **if** rows(B) $= 2\ell$ **then**
5: $[B;$ $;V] = \mathrm{DenseShrinkR}(B)$
▶ *Here* $\ell = 3k$, *thus* $B = \mathrm{FD}(A; \frac{1}{2k}; k)$
6: $F = F + k$ k_F^2,
▶ *### Next: subsample* $C :=$ V, *and then compressed the sampled rows using standard* FD
7: **for** $j = 1$ **to** 2ℓ **do**
8: $p_j = \frac{\Sigma_j^2}{\alpha^2 F}$
9: Sample C_j with probability p_j.
10: **if** C_j is sampled **then**
11: Set $\mathbf{v} = \frac{C_j}{\sqrt{p_j}}$.
12: $Q = \mathrm{FDShrink}([Q : \mathbf{v}]; \frac{1}{\alpha})$
▶ *Invoking* FD *with* $k = 0$
13: **end if**
14: **end for**
15: **end if**
16: **end for**
17: **return** $[B; Q]$

Correctness. We note that, at the end of Algorithm 4, $B = \mathrm{FD}(A; \frac{1}{2k}; k)$, so kA^TA B^TBk_2 kA $[A]_k k_F^2$=2k; or equivalently

$$\max_{x:\|x\|=1} j\,kAxk^2 \quad kBxk^2\,j \quad kA \quad [A]_k k_F^2 = 2k; \qquad (3)$$

Let $^{(i)}$, $V^{(i)}$, and $B^{(i)}$ be the value of $$, V, and B respectively returned by ith DenseShrinkR operation (line 5). Let

$C^{(i)} = {}^{(i)}V^{(i)}$. We use $B'^{(i)}$ to denote the value of B right before the ith DenseShrinkR operation (or the input of the ith DenseShrinkR operation) . From Algorithm 2, we have that

$$B'^{(i)T}B'^{(i)} = B^{(i)T}B^{(i)} + V^{(i)T} \quad {}^{(i)2}V^{(i)}$$
$$= B^{(i)T}B^{(i)} + C^{(i)T}C^{(i)};$$

Let $A^{(i)}$ be the rows of A arrived between the $(i \quad 1)$th and the ith DenseShrinkR operation, which means $B'^{(i)} = [B^{(i-1)}; A^{(i)}]$, and thus

$$B'^{(i)T}B'^{(i)} = B^{(i-1)T}B^{(i-1)} + A^{(i)T}A^{(i)};$$

Combined with the previous equality, we get

$$A^{(i)T}A^{(i)} + B^{(i-1)T}B^{(i-1)} \quad B^{(i)T}B^{(i)} = C^{(i)T}C^{(i)};$$

Let t be the total number of iterations. We define $B^{(0)} = 0$, and $C = [C^{(1)}; \quad ; C^{(t)}]$. Summing the above equality over $i = 1; \quad ; t$, we have

$$C^TC = \sum_i C^{(i)T}C^{(i)}$$
$$= \sum_i \left(A^{(i)T}A^{(i)} + B^{(i-1)T}B^{(i-1)} \quad B^{(i)T}B^{(i)}\right)$$
$$= A^TA \quad B^TB;$$

It follows that

$$\|C\|_F^2 = \text{trace}(C^TC) = \text{trace}(A^TA) \quad \text{trace}(B^TB)$$
$$= \|A\|_F^2 \quad \|B\|_F^2;$$

Now we bound $\|A\|_F^2 \quad \|B\|_F^2$ using similar ideas as in (Ghashami & Phillips, 2014). Let w_j be the jth singular vector of A, we have

$$\|C\|_F^2 = \|A\|_F^2 \quad \|B\|_F^2$$
$$= \sum_{j=1}^{k} \|Aw_j\|^2 + \sum_{j=k+1}^{d} \|Aw_j\|^2 \quad \|B\|_F^2$$
$$\sum_{j=1}^{k} \|Aw_j\|^2 + \|A\| \quad \|A\|_k\|_F^2 \quad \sum_{j=1}^{k} \|Bw_j\|^2$$

because $\sum_{j=1}^{k} \|Bw_j\|^2 \quad \|B\|_F^2$

$$\|A\| \quad \|A\|_k\|_F^2 + k \quad \|A\| \quad \|A\|_k\|_F^2/2k \quad \text{by Eq (3)}$$
$$= 1.5\|A\| \quad \|A\|_k\|_F^2; \tag{4}$$

In the algorithm, each row of C is sampled with probability $\frac{\|C_j\|^2}{\alpha^2 F}$, where F is the current squared F-norm of C. Let C_s be the sampled rows. Given Eq (4), we can prove the following using Theorem 2

$$\|C^TC \quad C_s^TC_s\|_2 \quad \|A\| \quad \|A\|_k\|_F^2, \text{ and}$$

$$\|C_s\|_F^2 = O(1) \quad \|A\| \quad \|A\|_k\|_F^2;$$

At the end of the algorithm, $Q = \text{FD}(C_s; \quad ; 0)$, then

$$\|C_s^TC_s \quad Q^TQ\|_2 \quad \|C_s\|_F^2 \quad O(\quad) \quad \|A\| \quad \|A\|_k\|_F^2;$$

Applying triangle inequality, we have $\|C^TC \quad Q^TQ\|_2$ $O(\quad) \quad \|A\| \quad \|A\|_k\|_F^2$, and thus

$$\|A^TA \quad B^TB \quad Q^TQ\|_2 = \|C^TC \quad Q^TQ\|_2$$
$$O(\quad) \quad \|A\| \quad \|A\|_k\|_F^2;$$

which proves the correctness.

Space and running time. The space is dominated by maintaining $B = \text{FD}(A; 1=2k; k)$ and $Q = \text{FD}(C_s; \quad ; 0)$, which is $O(dk + d= \quad)$ in total.

The running time of computing B is $O(ndk)$, and the running time for Q is $O(\text{rows}(C_s)d= \quad)$FD bound

4. The final sketch is $S = [B; C]$.

Note that $S = [B; C]$ approximate $[A'; R]$ in the sense that

$$kA'^T A' + R^T R \quad B^T B \quad C^T C k_2$$
$$kA'^T A' \quad B^T B k_2 + k R^T R \quad C^T C k_2$$
$$O(\ ) \ kA \quad [A]_k k_F^2:$$

From step (1), we have $A^T A = A'^T A' + R^T R$, and thus $[B; C]$ is a good approximation of A. Next we briefly describe how to implement this in one pass and small space.

To achieve (1), we use the following new idea. Let $Z \ 2 \ \mathbb{R}^{O(k)\times d}$ be an orthonormal matrix satisfying kA $Z^T Z A k_F^2$ $O(1)kA$ $[A]_k k_F^2$: Let $A' = ZA$ and $R = (I \quad Z^T Z)A$. It is easy to check that A' and R satisfy the requirement of (1). In the streaming model, we divide A into blocks, each of which contains roughly dk non-zero entries, and thus there are at most $t = \frac{nnz(A)}{dk}$ blocks. We use the above idea for each of the blocks and concatenate the results together. More precisely, for each block $A^{(i)} \ 2 \ \mathbb{R}^{\ell_i \times d}$, we use an input-sparsity time algorith9.96(0 Td [(;)]0 Td [(;)]0 T-331(64(,)-2m)-250(/F1962/F11r002n/F1962/F11Ral)-2 T-33)-250(final)- 7.9892 T5(this)T-3comme

Theorem 7. *Given any matrix $A \in \mathbb{R}^{n\times d}$, $0 < \varepsilon < 1$ and $0 < k \le d$, there is a streaming algorithm which maintains a strong $(\varepsilon; k)$-*proj*-sketch $S \in \mathbb{R}^{O(k/\varepsilon)\times d}$. The algorithm uses $O(dk/\varepsilon)$ space and runs in $O(\mathrm{nnz}(A)k + \mathrm{nnz}(A)\log n) + O(nk^3 + d \cdot \mathrm{poly}(k\varepsilon^{-1}))$.*

4.3. Proof of Theorem 6

Proof of Theorem 6. The detail of our fast algorithm for sparse matrix is described in Algorithm 5.

We let $A' = [A'^{(1)}; \cdots; A'^{(t)}]$, $R = [R^{(1)}; \cdots; R^{(t)}]$, and $Q = [Q^{(1)}; \cdots; Q^{(t)}]$. We use $w^{(i)}$ to denote the vector w in ith iteration. We need some technical lemmas.

Lemma 2. *With probability at least $1 - 1/n$, (1) $\|A' - [A']_k\|_F \le \|A - [A]_k\|_F$; (2) $\|R\|_F^2 \le O(1) \cdot \|A - [A]_k\|_F^2$.*

Proof. We divide A and A' into blocks as defined in Algorithm 5, i.e. $A = [A^{(1)}; \cdots; A^{(t)}]$ and $A' = [A'^{(1)}; \cdots; A^{(t)}]$. For each i, we have $A'^{(i)} = Z^{(i)}A^{(i)}$ for some matrix $Z^{(i)}$ with $O(k)$ orthonormal rows such that, with probability at least $1 - 1/n^2$,

$$\|A^{(i)} - Z^{(i)T}Z^{(i)}A^{(i)}\|_F^2 \le O(1) \cdot \|A^{(i)} - [A^{(i)}]_k\|_F^2. \quad (5)$$

By union bound, with probability at least $1 - 1/n$, eqn (5) holds for all i simultaneously. Let P be the projection matrix onto the subspace spanned by the top-k right singular vectors of A. So we have

$$\begin{aligned}
&\|A' - [A']_k\|_F^2 \le \|A' - A'P\|_F^2 \\
&= \sum_{i=1}^{t} \|A'^{(i)} - A'^{(i)}P\|_F^2 && P \text{ has rank } k \\
&= \sum_{i=1}^{t} \|Z^{(i)}A^{(i)} - Z^{(i)}A^{(i)}P\|_F^2 \\
&\le \sum_{i=1}^{t} \|A^{(i)} - A^{(i)}P\|_F^2 && Z^{(i)} \text{ is a orthogonal} \\
&= \|A - AP\|_F^2 = \|A - [A]_k\|_F^2; && \text{by definition of } P
\end{aligned}$$

which proves (1).

As defined in Algorithm 5, $R^{(i)} = (I - Z^{(i)T}Z^{(i)})A^{(i)}$, where $Z^{(i)}$ satisfies (5). Therefore,

$$\begin{aligned}
\|R\|_F^2 &= \sum_{i=1}^{t} \|(I - Z^{(i)T}Z^{(i)})A^{(i)}\|_F^2 \\
&\le O(1) \cdot \sum_{i=1}^{t} \|A^{(i)} - [A^{(i)}]_k\|_F^2 \\
&= O(1) \cdot \|A - [A]_k\|_F^2
\end{aligned}$$

which proves (2). □

Lemma 3. *$A^TA = R^TR + A'^TA'$; with probability $1 - 1/n$ it holds that $\|A^TA - A'^TA'\|_F \le O(1) \cdot \|A - [A]_k\|_F^2$.*

Proof. To prove the first part, we only need to prove $A^{(i)T}A^{(i)} = R^{(i)T}R^{(i)} + A'^{(i)T}A'^{(i)}$ holds for all i. Recall that $A'^{(i)} = Z^{(i)}A^{(i)}$. For each i, we have

$$\begin{aligned}
R^{(i)T}R^{(i)} &= A^{(i)T}(I - Z^{(i)T}Z^{(i)})(I - Z^{(i)T}Z^{(i)})A^{(i)} \\
&= A^{(i)T}(I - Z^{(i)T}Z^{(i)})A^{(i)} \\
&= A^{(i)T}A^{(i)} - A'^{(i)T}A'^{(i)}.
\end{aligned}$$

This proves the first part, from which, we also get

$$\|A^TA - A'^TA'\|_F = \|R^TR\|_F \le \|R\|_F^2,$$

where the inequality is from the submultiplicative of matrix norms. Then the second part follows from Lemma 2. □

Lemma 4. *If the entries of A are integers bounded in magnitude by $\mathrm{poly}(nd)$ and $\mathrm{rank}(A) \ge 1$, then $\|A - [A]_k\|_F^2 \ge 1/\mathrm{poly}(nd)$.*

Proof. The lemma directly follows from a result of (Clark-

entry of A is integer bounded in magnitude by poly(nd), which implies the number of epochs is

$$O(\log \frac{kRk_F^2}{\quad}) \quad O\left(\log\left(kAk_F^2 \quad \text{poly}(nd)\right)\right) = O(\log(nd)):$$

The number of rows sampled in each epoch is at most $O(1= \ ^2)$: let $a_1; \quad ; a_t$ be the rows of R in the epoch, and thus $\sum_j ka_jk^2 \quad O(F)$ (otherwise the epoch ends); each row a_j is sampled with probability $(\frac{\|a_j\|^2}{\alpha^2 F})$, which implies the total number of rows sampled is $O(1= \ ^2)$. This proves the third part.

The case $\text{rank}(A) \quad 1:1k$ is easier: we can set the rank parameter k a little larger (say $k' = 2k$) in our algorithm so that R is always $\mathbf{0}$ by Lemma 2, and thus $\text{rows}(Q) = 0$. In this case, our algorithm is essentially exact. □

Correctness. By union bound, with probability 0:9 all the above lemmas hold simultaneously, and we will assume this happens. Since $C = \text{FD}(Q; \ ; 0)$, by Theorem 1 and Lemma 6, we have

$$kQ^TQ \quad C^TCk_2 \qquad kQk_F^2 \quad O(\ ) \ kRk_F^2: \qquad (6)$$

Since B is a matrix such that $B = \text{FFDdense}(A'; \ ; k)$, by Theorem 5, we have

$$kA'^TA' \quad B^TBk_2 \qquad kA' \quad [A']_kk_F^2 \qquad kA \quad [A]_kk_F^2; \qquad (7)$$

where the last inequality is from Lemma 2. Let $S = [B; C]$,

$$\begin{aligned}
kA^TA \quad S^TSk_2 &= kA^TA \quad B^TB \quad C^TCk_2 \\
&\quad kA^TA \quad A'^TA' \quad C^TCk_2 + kA'^TA' \quad B^TBk_2 \\
&\quad kA^TA \quad A'^TA' \quad C^TCk_2 + \quad kA \quad [A]_kk_F^2 \quad \text{by (7)} \\
&= kR^TR \quad C^TCk_2 + \quad kA \quad [A]_kk_F^2 \\
&\quad kR^TR \quad Q^TQk_2 + kQ^TQ \quad C^TCk_2 + \quad kA \quad [A]_kk_F^2 \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{triangle inequality} \\
&\quad O(\ ) \ kA \quad [A]_kk_F^2 + O(\ ) \ kRk_F^2 + \quad kA \quad [A]_kk_F^2 \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{by (6) and Lemma 6} \\
&\quad O(\ ) \ kA \quad [A]_kk_F^2; \qquad\qquad\qquad\qquad\qquad\qquad \text{by Lemma 2}
\end{aligned}$$

which proves the error bound after adjusting by a constant.

Space and running time. For space, we need a buffer to store a new block of A, the size of which is at most $dk + d$, as the $\text{nnz}(A^{(i)})$ is at most $dk + d$. When applying Theorem 3, we set $= 1=n^2$, and the input matrix has at most $\frac{d}{k\log(nd)}$ rows, so we need $O(dk)$ space to compute and store Z. $A'^{(i)}$ is of dimension $O(k) \quad d$, which needs $O(dk)$ space to compute and store. To naively compute $W = (I \quad Z^TZ)A^{(i)}$, we need $O(dk + d\log n)$ space. However, observe that we do not have to compute W explicitly, we only need to know its row norms, i.e. the vector w. To save space, we compute one column of W at a time, i.e., generate columns of one by one, and update w iteratively, and thus the extra space used is $O(d)$. From Theorem 5, the space used by $\text{FFDdense}(A'; \ ; k)$ is $O(d(k + \ ^{-1}))$. Note that, in line 11 of Algorithm 5, the rows of $Q^{(i)}$ can be computed one by one, and thus compute $C = \text{FD}(Q; \ ; 0)$ uses $O(d= \ )$ space by theorem 1. So the total space usage is bounded by $O(d(k + \ ^{-1}))$.

Let $`_i$ be the number of rows in $i$th block $A^{(i)}$. The time to compute $Z$ using Theorem 3 is $O(\text{nnz}(A^{(i)}\log n) + O(`_ik^3)$. Hence the total time used on this step is

$$\sum_i O(\text{nnz}(A^{(i)}\log n) + O(`_ik^3) = O(\text{nnz}(A)\log n) + O(nk^3):$$

The step to compute the matrix multiplication $A'^{(i)} = ZA^{(i)}$ takes $O(\text{nnz}(A^{(i)})k)$ time, since Z has $O(k)$ rows. So the total time spent on this step is $O(\text{nnz}(A)k)$. By the definition of blocks, there are at most $O(\frac{\text{nnz}(A)}{dk} + \frac{nk\log(nd)}{d})$ blocks. After left multiplying Z, each block contributes $O(k)$ rows to A', and thus the total number of rows in A' is at most $O(\frac{\text{nnz}(A)}{d} + \frac{nk^2\log(nd)}{d})$. Computing B by invoking $B = \text{FFDdense}(A'; \ ; k)$ needs $O(\text{rows}(A')dk) + O(d= \ ^3) = O(\text{nnz}(A)k) + O(nk^3 + d= \ ^3)$ time. Finally, each row of Q can be computed in time $O(dk)$ given A' (which has been computed in line 5). Invoking $C = \text{FD}(Q; \ ; 0)$ needs $O(d= \ ^3 + dk= \ ^2)$ since $\text{rows}(Q) = O(1= \ ^2)$ by Lemma 6. The total time is thus $O(\text{nnz}(A)k + \text{nnz}(A)\log n) + O(nk^3 + d \ ^{-3} + dk \ ^{-2})$. □

5. Conclusion

In this paper, we study covariance sketches for matrices in the streaming model. We provide new space-optimal algorithms with improved running time. We also prove that our running times cannot be significantly improved unless the state-of-the-art matrix multiplication algorithms can. Thus, we almost settle the time complexity of this problem.

Acknowledgments

This work is supported by Shanghai Science and Technology Commission (Grant No. 17JC1420200) and Shanghai Sailing Program (Grant No. 18YF1401200).

References

Avron, Haim, Sindhwani, Vikas, and Woodruff, David. Sketching structured matrices for faster nonlinear regression. In *Advances in neural information processing systems*, pp. 2994–3002, 2013.

Boutsidis, Christos, Garber, Dan, Karnin, Zohar, and Liberty, Edo. Online principal components analysis. In *SODA*. SIAM, 2015.

Boutsidis, Christos, Woodruff, D, and Zhong, Peilin. Optimal principal component analysis in distributed and streaming models. *STOC*, 2016.

Calandriello, Daniele, Lazaric, Alessandro, and Valko, Michal. Efficient second-order online kernel learning with adaptive embedding. In *Advances in Neural Information Processing Systems*, pp. 6142–6151, 2017.

Chierichetti, Flavio, Gollapudi, Sreenivas, Kumar, Ravi, Lattanzi, Silvio, Panigrahy, Rina, and Woodruff, David. Algorithms for ℓ_p low rank approximation. In *ICML*, 2017.

Clarkson, Kenneth L and Woodruff, David P. Numerical linear algebra in the streaming model. In *STOC*, pp. 205–214. ACM, 2009.

Clarkson, Kenneth L and Woodruff, David P. Low rank approximation and regression in input sparsity time. In *STOC*, 2013.

Drineas, Petros, Kannan, Ravi, and Mahoney, Michael W. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

Ghashami, Mina and Phillips, Jeff M. Relative errors for deterministic low-rank matrix approximations. In *SODA*, pp. 707–717. SIAM, 2014.

Ghashami, Mina, Phillips, Jeff M, and Li, Feifei. Continuous matrix approximation on distributed data. *Proceedings of the VLDB Endowment*, 7(10):809–820, 2014.

Ghashami, Mina, Liberty, Edo, and Phillips, Jeff M. Efficient frequent directions algorithm for sparse matrices. *KDD*, 2016.

Huang, Hao and Kasiviswanathan, Shiva Prasad. Streaming anomaly detection using randomized matrix sketching. *Proceedings of the VLDB Endowment*, 9(3):192–203, 2015.

Huang, Zengfeng, Lin, Xuemin, Zhang, Wenjie, and Zhang, Ying. Efficient matrix sketching over distributed data. In *Proceedings of PODS*, pp. 347–359. ACM, 2017.

Johnson, William B and Lindenstrauss, Joram. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Kannan, Ravi, Vempala, Santosh, and Woodruff, David. Principal component analysis and higher correlations for distributed data. In *Proceedings of The 27th Conference on Learning Theory*, pp. 1040–1057, 2014.

Karnin, Zohar and Liberty, Edo. Online pca with spectral bounds. In *Conference on Learning Theory*, pp. 1129–1140, 2015.

Kim, Young-Bum, Stratos, Karl, and Sarikaya, Ruhi. Scalable semi-supervised query classification using matrix sketching. In *ACL*, volume 2, pp. 8–13, 2016.

Leng, Cong, Wu, Jiaxiang, Cheng, Jian, Bai, Xiao, and Lu, Hanqing. Online sketching hashing. In *IEEE CVPR*, pp. 2503–2511, 2015.

Liang, Yingyu, Balcan, Maria-Florina F, Kanchanapally, Vandana, and Woodruff, David. Improved distributed principal component analysis. In *Advances in Neural Information Processing Systems*, pp. 3113–3121, 2014.

Liberty, Edo. Simple and deterministic matrix sketching. In *KDD*, pp. 581–588. ACM, 2013.

Luo, Haipeng, Agarwal, Alekh, Cesa-Bianchi, Nicolo, and Langford, John. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems*, pp. 902–910, 2016.

Musco, Cameron and Musco, Christopher. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, pp. 1396–1404, 2015.

Musco, Cameron and Woodruff, David. Is input sparsity time possible for kernel low-rank approximation? In *Advances in Neural Information Processing Systems*, pp. 4438–4448, 2017.

Sarlos, Tamas. Improved approximation algorithms for large matrices via random projections. In *FOCS*, 2006.

Song, Qiang, Cheng, Jian, and Lu, Hanqing. Incremental matrix factorization via feature space re-learning for recommender system. In *RecSys*, pp. 277–280. ACM, 2015.

Wei, Zhewei, Liu, Xuancheng, Li, Feifei, Shang, Shuo, Du, Xiaoyong, and Wen, Ji-Rong. Matrix sketching over sliding windows. *SIGMOD*, 2016.

Woodruff, David. Low rank approximation lower bounds in row-update streams. In *Advances in Neural Information Processing Systems*, pp. 1781–1789, 2014.

Ye, Qiaomin, Luo, Luo, and Zhang, Zhihua. Frequent direction algorithms for approximate matrix multiplication with applications in cca. In *AAAI*, pp. 2301–2307. AAAI Press, 2016.

Zhang, Yuchen, Wainwright, Martin, and Jordan, Michael. Distributed estimation of generalized matrix rank: Efficient algorithms and lower bounds. In *Proceedings of*

the 32nd International Conference on Machine Learning (ICML-15), pp. 457–465, 2015.