Journal of Machine Learning Research 20 (2019) 1-23 Submitted 12/18; Published 2/19

Near Optimal Frequent Directions for Sketching Dense and Sparse Matrices

Zengfeng Huang HUANGZF@FUDAN.EDU.CN
School of Data Science
Fudan University
Shanghai, China

Editor: Kilian Weinberger

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 consider the problems in the streaming model, where the algorithm can only make one pass over the input with limited working space, and we are interested in minimizing the *covariance error* $\|A^TA - B^TB\|_2$. The popular Frequent Directions algorithm of Liberty (2013) and its variants achieve optimal space-error tradeoffs. However, whether the running time can be improved remains an unanswered question. In this paper, we almost settle the question by proving that the time complexity of this problem is equivalent to that of matrix multiplication up to lower order terms. Specifically, we provide new space-optimal algorithms with faster running times and also show that the running times of our algorithms can be improved if and only if the state-of-the-art running time of matrix multiplication can be improved significantly.

Keywords: Matrix Sketching, Frequent Directions, Streaming Algorithms

1. Introduction

For large-scale matrix computations, exact algorithms are often too slow, so a large body of work focuses on designing fast randomized approximation algorithms. Matrix sketching is a commonly used algorithmic technique for solving linear algebra problems over massive data matrices, e.g., Sarlos (2006); Clarkson and Woodruff (2013); Avron et al. (2013); Chierichetti et al. (2017). In the sketch-and-solve framework, given a large matrix, we first compute a small sketch using a lightweight algorithm, and then do more expensive computations on the sketch instead of on the original matrix. Thus, the key is how to efficiently compute sketch matrices that are small but still preserve vital properties of the input matrix.

In real-world applications, data often arrives in a streaming fashion and it is often impractical or impossible to store the entire data set in the main memory. This paper studies 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; algorithm is only allowed to make one pass over the stream with severely limited working space, and is required to maintain a sketch continuously. This problem has received lots of attention recently (Liberty, 2013; Ghashami and Phillips, 2014; Woodruff, 2014a; Ghashami et al., 2016; Wei et al., 2016).

The popular *Frequent Directions* algorithms (Liberty, 2013; Ghashami et al., 2016) achieve optimal tradeoffs between space usage and approximation error (Woodruff, 2014a), which have found lots of applications in online learning, e.g., Boutsidis et al. (2015); Karnin and Liberty (2015); Leng et al. (2015); Huang and Kasiviswanathan (2015); Luo et al. (2016); Calandriello et al. (2017), and other important problems (Song et al., 2015; Ye et al., 2016; Kim et al., 2016; Cohen et al., 2017). However, in contrast to the space complexity, 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 (see e.g., Clarkson and 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 $A \in \mathbb{R}^{n\times d}$, the problem is 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 $0 < \alpha < 1$, and integer $0 \le k \le \mathsf{rank}(A)$, we will call B an (α, k)-*cov*-sketch of A, if the covariance error*[1]

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

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 and Phillips (2014) that one can obtain relative projection error from small covariance error. We include a proof of the next lemma in Appendix A.1.

Lemma 3 (Covariance Error to Projection Error (Ghashami and Phillips, 2014))

$$\|A - \pi_B^k(A)\|_F^2 \le \|A - [A]_k\|_F^2 + 2k\cdot\|A^TA - B^TB\|_2.$$

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

$$\|A - \pi_B^k(A)\|_F^2 \le (1+\varepsilon)\|A - [A]_k\|_F^2. \tag{2}$$

We will often refer to $(\frac{\varepsilon}{2k}, k)$-cov-sketches as (ε, k)-proj-sketches.

(ε, k)-proj-sketches actually satisfy a more general property, called *Projection-Cost Preserving* property. Specially, suppose B is an (ε, k)-proj-sketch of a matrix A, then

$$\|B - BQQ^T\|_F^2 = (1\pm\varepsilon)\|A - AQQ^T\|_F^2,$$

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

for all rank-k orthonormal matrices Q (see Feldman et al. (2013); Cohen et al. (2015, 2017) for more details). One immediate application of projection-cost preserving sketches is constrained low rank approximations, which can be formulated as the following optimization problem:

$$\min_{X:\mathsf{rank}(X)\leq k, X\in\mathcal{S}} \|A - AXX^T\|_F^2,$$

where $\mathcal{S}$ is some constraint; special cases include sparse PCA, nonnegative PCA, and k-means clustering (Feldman et al., 2013; Cohen et al., 2015). Given any projection-cost preserving sketch B of A, one can solve the above optimization problem with respect to B to obtain an approximate solution, which significantly reduces the computational costs when B is much smaller than A.

Modern data matrices are often large and sparse, so we will always assume n and d are very large and $\mathsf{nnz}(A) \ll nd$, where $\mathsf{nnz}(A)$ 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 Directions* (FD) algorithm (Liberty, 2013), with an improved analysis by Ghashami and Phillips (2014), maintains an (α, k)-cov-sketch $B \in \mathbb{R}^{\ell\times d}$ at any time, where $\ell = O(k + \alpha^{-1})$. 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 $O(\mathsf{nnz}(A)\ell \log d + \mathsf{nnz}(A)\log n + n\ell^2)$ by Ghashami et al. (2016). Setting $\alpha = \varepsilon/2k$ (or $\ell = O(k/\varepsilon)$) and by Lemma 3, B is a (ε, k)-proj-sketch. Now, B contains $O(k/\varepsilon)$ rows, and the space and running time become $O(dk/\varepsilon)$ and $O(\mathsf{nnz}(A)k\varepsilon^{-1}\cdot\log d + \mathsf{nnz}(A)\log n + nk^2\varepsilon^{-2})$ respectively. The $\log d$ factor in the leading term was removed by Luo et al. (2016). It was shown by Woodruff (Woodruff, 2014a) 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 $O(\mathsf{nnz}(A)+(n+d)\cdot\mathrm{poly}(k\alpha^{-1}))$, 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 give a new space-optimal streaming algorithm with $O(ndk) + \tilde{O}(d\alpha^{-3})$ running time to compute (α, k)-cov-sketches for dense matrices, which improves the original FD algorithm for small α. The running time is optimal up to lower order terms, provided matrix multiplication cannot be improved significantly.

2. Based on our fast algorithm for sketching dense matrices, we then give a new space-optimal streaming algorithm with $O(\mathsf{nnz}(A)k + nk^3) + \tilde{O}(d\alpha^{-3})$ running time to compute (α, k)-cov-sketches for sparse matrices. We separate the dependence of

	Time	
	(α, k)-cov	(ε, k)-proj
FD (Liberty, 2013)	$O(ndk + nd/\alpha)$	$O(ndk/\varepsilon)$
FFDdense (**new**)	$O(ndk)$	$O(ndk)$
Sparse FD(Ghashami et al., 2016)	$O(\mathsf{nnz}(A)k \log d + \mathsf{nnz}(A) \log d/\alpha)$	$O(\mathsf{nnz}(A)k \log d/\varepsilon)$
FFDsparse (**new**)	$O(\mathsf{nnz}(A)k)$	$O(\mathsf{nnz}(A)k)$
Lower bounds (**new**)	$\Omega(\mathsf{nnz}(A)k)$	$\Omega(\mathsf{nnz}(A)k)$

Table 1: Running times and lower bounds for streaming (α, k)-cov-sketch and (ε, k)-proj-sketch algorithms. Lower order terms are omitted and the lower bounds are conditional.

$1/\alpha$ from $\mathsf{nnz}(A)$, which improves the results of Ghashami et al. (2016) for small α. Therefore, to compute an (ε, k)-proj-sketch, our algorithm only needs $O(\mathsf{nnz}(A)k)$ time (ignoring lower order terms) as opposed to $O(\mathsf{nnz}(A)k\varepsilon^{-1} \cdot \log d)$ in Ghashami et al. (2016) (see Table 1).

3. We show that $o(\mathsf{nnz}(A)k)$ time is likely very difficult to achieve, as it will imply a breakthrough in fast matrix multiplication. Specifically, we prove that, under mild assumptions, the time complexity for computing an $(O(1), k)$-cov-sketch $B \in \mathbb{R}^{O(k)\times d}$ of A in the streaming model is equivalent to the time complexity of left multiplying A by an arbitrary matrix $C \in \mathbb{R}^{k\times n}$.

1.4. Other Related Work

The problem of computing (α, 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; Zhang 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 and 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 $\mathsf{Diag}(x) \in \mathbb{R}^{d\times d}$ is a diagonal matrix such that the ith diagonal entry is x_i. For a matrix $A \in \mathbb{R}^{n\times d}$ 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. $\mathsf{nnz}(A)$ is the number of non-zero entries in A and $\mathsf{rows}(A)$ is the number of rows in A. We write the (reduced) singular value decomposition of A as $(U, \Sigma, V) = \mathsf{SVD}(A)$. 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 $k \leq \mathsf{rank}(A)$, 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 $\tilde{O}()$ to hide polylog(ndk) factors.

1.6. Tools

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

Theorem 4 (Liberty (2013)) *Given $A \in \mathbb{R}^{n\times d}$, in one pass, $\mathsf{FD}(A, \alpha, k)$ processes A in $O(nd(k + \alpha^{-1}))$ time and $O(d(k + \alpha^{-1}))$ space. It maintains a matrix $B \in \mathbb{R}^{O(k+\alpha^{-1})\times d}$ such that $\|A^TA - B^TB\|_2 \le \alpha\|A - [A]_k\|_F^2$.*

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 *sampling with replacement*, 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 sampling with replacement, which can be found in Appendix.

Theorem 5 *For any $A \in \mathbb{R}^{n\times d}$ and $F > 0$, we sample each row A_i with probability $p_i \ge \frac{\|A_i\|^2}{\alpha^2 F}$; if it is sampled, scale it by $1/\sqrt{p_i}$. Let B be the (rescaled) sampled rows, then w.p. 0.99, $\|A^TA - B^TB\|_2 \le 10\alpha\sqrt{F}\|A\|_F$ and $\|B\|_F \le 10\|A\|_F$. The expected number of rows sampled is $O(\frac{\|A\|_F^2}{\alpha^2 F})$ if $p_i = O(\frac{\|A_i\|^2}{\alpha^2 F})$ for each i.*

Input-sparsity time subspace embedding. We will use fast subspace embeddings (Clarkson and Woodruff, 2013) to speedup computation. There are various approaches to achieve input-sparsity time subspace embedding with different guarantees; the following result suffices for our purpose, the proof of which can be found in Woodruff (2014b) (Lemma 4.2 and Remark 4.1).

Theorem 6 (Subspace Embedding) *Given any $A \in \mathbb{R}^{n\times d}$, there exist (random) matrices $J \in \mathbb{R}^{O(k)\times t_1}$ and $C \in \mathbb{R}^{t_1\times d}$, with $t_1 = \min(d, O(k^2))$, such that, with probability 0.99, the column space of $S = AC^TJ^T$ contains an $O(1)$-approximate rank-k approximation to A. More precisely, there exists X with $\mathsf{rank}(X) \le k$ such that*

$$\|A - SX\|_F^2 \le O(1)\|A - [A]_k\|_F^2,$$

Moreover, C only contains $O(d)$ nonzero entries and $S = AC^TJ^T$ can be computed in time $O(\mathsf{nnz}(A)) + O(nk^3)$.

One can choose J to be a matrix with iid Gaussian random variables and C to be a count-sketch matrix (Clarkson and Woodruff, 2013)

2. Algorithm for Dense Matrices

Theorem 7 ($\mathsf{FFDdense}(A, \alpha, k)$) *Given a matrix $A \in \mathbb{R}^{n\times d}$, $0 < \alpha < 1$ and $0 \le k \le d$, $\mathsf{FFDdense}(A, \alpha, k)$ processes A in one pass using $O(ndk) + \tilde{O}(d\alpha^{-3})$ time and $O(dk + d\alpha^{-1}))$ space, and maintains a matrix $B^{O(k+\alpha^{-1})\times d}$. With probability 0.99, it holds that $\|A^TA - B^TB\|_2 \le \alpha\|A - [A]_k\|_F^2$.*

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 $\ell \leq d$, 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 $\mathsf{FDShrink}(B, v, \ell)$ (Algorithm 3). In this procedure, we first append a after B; if B has no more than 2ℓ rows we do nothing, and otherwise do a DenseShrink operation (Algorithm 1) on B, which halves the number of rows in B (after removing zero rows). It was proved by Liberty (2013) and Ghashami and Phillips (2014) that for $\ell = k + \alpha^{-1}$, we have

$$\|A^T A - B^T B\|_2 \leq \alpha \|A - [A]_k\|_F^2.$$

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 $O(nd\ell) = O(nd(k + \alpha^{-1}))$. Our goal is to separate nd from α^{-1} in the running time.

Algorithm 1 DenseShrink

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

1: Compute $[U, \Sigma, V] = \mathsf{SVD}(B)$, and $\sigma = \Sigma_{\ell,\ell}$.
2: $\hat{\Sigma} = \sqrt{\max(\Sigma^2 - \sigma^2 I_{2\ell}, 0)}$
3: **return** $B = \hat{\Sigma} V^T$

Algorithm 2 DenseShrinkR

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

1: Compute $[U, \Sigma, V] = \mathsf{SVD}(B)$, and $\sigma = \Sigma_{\ell,\ell}$.
2: $\hat{\Sigma} = \sqrt{\max(\Sigma^2 - \sigma^2 I_{2\ell}, 0)}$
3: $\bar{\Sigma} = \sqrt{\Sigma^2 - \hat{\Sigma}^2}$ ▶ $\Sigma^2 = \bar{\Sigma}^2 + \hat{\Sigma}^2$
4: **return** $B = \hat{\Sigma} V^T$, $\bar{\Sigma}$ and V^T

Algorithm 3 FDShrink

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

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

To achieve this, we first compute a coarse approximation using FD by invoking $B = \mathsf{FD}(A, k, \frac{1}{2k})$, 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 $A^T A = B^T B + C^T C$ and $\|C\|_F^2 \leq \|A - [A]_k\|_F^2$. 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 4 FFDdense

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

1: $F = 0$, $\ell = 3k$, $Q =$ **empty**, $B =$ **empty**
2: **for** $i = 1$ **to** n **do**
3: Append A_i after B
4: **if** rows$(B) = 2\ell$ **then**
5: $[B, \Sigma, V] =$ DenseShrinkR(B) ▶ *Here $\ell = 3k$, thus $B = \mathsf{FD}(A, \frac{1}{2k}, k)$*
6: $F = F + \|\Sigma\|_F^2$, ▶ *Next: subsample $C := \Sigma V$, and then compress 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: $Q = \mathsf{FDShrink}([Q; \mathbf{v}], \frac{1}{\alpha})$ ▶ *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, $B = \mathsf{FD}(A, \frac{1}{2k}, k)$, so $\|A^TA - B^TB\|_2 \le \|A - [A]_k\|_F^2/2k$, or equivalently

$$\max_{x:\|x\|=1} \left| \|Ax\|^2 - \|Bx\|^2 \right| \le \|A - [A]_k\|_F^2/2k. \tag{3}$$

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

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

Let $A^{(i)}$ be the rows of A arrived between the $(i-1)$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

$$A^{(i)T}A^{(i)} + B^{(i-1)T}B^{(i-1)} - B^{(i)T}B^{(i)} = C^{(i)T}C^{(i)}.$$

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

$$\begin{aligned} C^TC = \sum_i C^{(i)T}C^{(i)} &= \sum_i \left(A^{(i)T}A^{(i)} + B^{(i-1)T}B^{(i-1)} - B^{(i)T}B^{(i)}\right) \\ &= A^TA - B^TB. \end{aligned}$$

It follows that

$$\|C\|_F^2 \quad = \quad \mathsf{trace}(C^TC) = \mathsf{trace}(A^TA) - \mathsf{trace}(B^TB) = \|A\|_F^2 - \|B\|_F^2.$$

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

$$\begin{aligned}
\|C\|_F^2 &= \|A\|_F^2 - \|B\|_F^2 \\
&= \sum_{j=1}^{k} \|Aw_j\|^2 + \sum_{j=k+1}^{d} \|Aw_j\|^2 - \|B\|_F^2 \\
&\leq \sum_{j=1}^{k} \|Aw_j\|^2 + \|A - [A]_k\|_F^2 - \sum_{j=1}^{k} \|Bw_j\|^2 \qquad \text{because } \sum_{j=1}^{k} \|Bw_j\|^2 \leq \|B\|_F^2 \\
&\leq \|A - [A]_k\|_F^2 + k \cdot \|A - [A]_k\|_F^2/2k \qquad \text{by Eq (3)} \\
&= 1.5\|A - [A]_k\|_F^2. \qquad (4)
\end{aligned}$$

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 5

$$\|C^TC - C_s^TC_s\|_2 \leq \alpha\|A - [A]_k\|_F^2, \text{ and } \|C_s\|_F^2 = O(1) \cdot \|A - [A]_k\|_F^2.$$

At the end of the algorithm, $Q = \mathsf{FD}(C_s, \alpha, 0)$, then

$$\|C_s^TC_s - Q^TQ\|_2 \leq \alpha\|C_s\|_F^2 \leq O(\alpha) \cdot \|A - [A]_k\|_F^2.$$

Applying triangle inequality, we have $\|C^TC - Q^TQ\|_2 \leq O(\alpha) \cdot \|A - [A]_k\|_F^2$, and thus

$$\|A^TA - B^TB - Q^TQ\|_2 = \|C^TC - Q^TQ\|_2 \leq O(\alpha) \cdot \|A - [A]_k\|_F^2,$$

which proves the correctness.

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

The running time of computing B is $O(ndk)$, and the running time for Q is $O(\mathsf{rows}(C_s)d/\alpha)$. To bound $\mathsf{rows}(C_s)$, we divide the stream into epochs, where F roughly doubles in each epoch. This means the total number of epochs is bounded by $O(\log(nd))$, since we assume each real number in the input can be represented by $O(\log(nd))$ bits. We emphasize that a rigorous analysis on this will be more subtle; see discussions in the proof of Lemma 15 below. Applying Theorem 5 on the submatrix in each epoch, it is easy to check the expected number of rows sampled in each epoch is $O(1/\alpha^2)$, so $\mathsf{rows}(C_s) = O(\frac{\log(nd)}{\alpha^2})$. Thus the total running time is $O(ndk) + \tilde{O}(d\alpha^{-3})$. We remark that the residual return by DenseShrinkR is in the form of $C = \Sigma V^T$, where Σ is diagonal and V has orthonormal columns. Therefore, the row norms of C are simply the diagonals of Σ.

3. Algorithm for Sparse Matrices

For sketching sparse matrices, we will use a subroutine (see Algorithm 5) for computing weak low rank approximations based on fast subspace embeddings. In this subroutine, the input matrix is an $\ell \times d$ matrix A and two additional matrices J and C, where J has $O(k)$ rows and C has d columns. In our applications, J, C will always be the subspace embedding matrices from Theorem 6. The algorithm outputs an $O(k) \times \ell$ matrix Z with orthonormal rows.

Algorithm 5 Weak Low Rank Approximation (LRA)

Input: $A \in \mathbb{R}^{\ell \times d}$, $J \in \mathbb{R}^{O(k) \times w}$ and $C \in \mathbb{R}^{w \times d}$

1: Compute $S = AC^T J^T$
2: Compute $Z \in \mathbb{R}^{O(k) \times \ell}$ whose rows form an orthonormal basis for the column space of S
3: **return** Z

The following lemma is a simple corollary of Theorem 6.

Lemma 8 *If J, C are the subspace embedding matrices from Theorem 6 with $w = \min(d, O(k^2))$ and Z is the output of Algorithm 5, then with probability 0.99*

$$\|A - Z^T ZA\|_F^2 \leq O(1)\|A - [A]_k\|_F^2.$$

Moreover, Z can be computed in $O(\mathsf{nnz}(A) + \ell k^3)$ time and $O(\ell k^2)$ extra space.

Proof From Theorem 6, there exists X with $\mathsf{rank}(X) \leq k$ such that

$$\|A - Z^T X\|_F^2 \leq O(1)\|A - [A]_k\|_F^2.$$

Hence,

$$\begin{aligned}\|A - Z^T ZA\|_F^2 &\leq \|A - Z^T ZA\|_F^2 + \|Z^T ZA - Z^T X\|_F^2 \\ &= \|A - Z^T X\|_F^2 \quad \text{Pythagorean theorem} \\ &\leq O(1)\|A - [A]_k\|_F^2.\end{aligned}$$

Note that S can be computed in time $O(\mathsf{nnz}(A) + \ell k^3)$ and Z can be computed from S in $O(\ell k^2)$ time. The space usage is dominated by storing the intermediate result AC^T, which is $O(\ell k^2)$. ■

3.1. Overview of Our Algorithm

Our approach is quite different from Ghashami et al. (2016). Their main idea is to use fast approximate SVD (Musco and Musco, 2015) in the original FD, which leads to suboptimal time. Our approach is summarized as follows.

1. Decompose $A^T A = A'^T A' + R^T R$, such that A' contains small number of rows and $\|A' - [A']_k\|_F^2 = O(1) \cdot \|A - [A]_k\|_F^2$. Moreover, $\|R\|_F^2 = O(1) \cdot \|A - [A]_k\|_F^2$.

2. Compute a sketch B of A' using fast FD algorithm for dense matrices (Theorem 7), which satisfies that $\|A'^T A' - B^T B\|_2 \le \alpha \|A' - [A']_k\|_F^2 \le \alpha \|A - [A]_k\|_F^2$.

3. Compute a sketch matrix C of R such that $\|R^T R - C^T C\|_2 \le \alpha \|R\|_F^2 \le O(\alpha) \cdot \|A - [A]_k\|_F^2$, which can be done via random sampling (Theorem 5) combined with FD.

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

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

$$\begin{aligned}\|A'^T A' + R^T R - B^T B - C^T C\|_2 &\le \|A'^T A' - B^T B\|_2 + \|R^T R - C^T C\|_2 \\ &\le O(\alpha) \cdot \|A - [A]_k\|_F^2.\end{aligned}$$

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 $Z \in \mathbb{R}^{O(k) \times d}$ be an orthonormal matrix satisfying $\|A - Z^T Z A\|_F^2 \le O(1)\|A - [A]_k\|_F^2$. Let $A' = ZA$ and $R = (I - Z^T Z)A$. 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 $A^{(i)} \in \mathbb{R}^{\ell_i \times d}$, we use an input-sparsity time algorithm (Algorithm 5 and Lemma 8) to compute a matrix $Z^{(i)} \in \mathbb{R}^{O(k) \times \ell_i}$ such that

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

Let $A'^{(i)} = Z^{(i)} A^{(i)}$, and $R^{(i)} = (I - Z^{(i)T} Z^{(i)}) A^{(i)}$. We then set $A' = [A'^{(1)}; \cdots; A'^{(t)}]$ and $R = [R^{(1)}; \cdots; R^{(t)}]$, and prove that A' and R satisfy the requirement of (1), where A' only has $t \times O(k) = O(\frac{\mathsf{nnz}(A)}{d})$ rows (since each block of A' has $O(k)$ rows). Here we do not compute R explicitly, as we will sample a subset of the rows from R. Note that the running time of this step is dominated by computing $Z^{(i)} A^{(i)}$, which is $O(\mathsf{nnz}(A^{(i)})k)$, and thus $O(\mathsf{nnz}(A)k)$ in total.

To compute B of step (2), we may use the standard $\mathsf{FD}(A', \alpha, k)$ (Theorem 4). Since A' has at most $O(\frac{\mathsf{nnz}(A)}{d})$ rows, B can be computed in $O(\mathsf{nnz}(A)(k + \alpha^{-1}))$ time. However, it still has an $\mathsf{nnz}(A)\alpha^{-1}$ term. So, we apply our faster FD algorithm for dense matrices (Theorem 7) on A', which only takes $O(\mathsf{nnz}(A)k) + \tilde{O}(d\alpha^{-3})$ time.

In order to compute a sketch C of R in step (3), we first subsample the rows of R using streaming Bernoulli sampling. One difficulty is that R could be dense, and it may take nd time to compute the row norms. Fortunately, each $R^{(i)}$ is of special form, and we are able to design a fast algorithm to compute its row norms (Algorithm 7). Let Q be the sampled rows, with $\mathsf{rows}(Q) = \tilde{O}(1/\alpha^2)$, and note that each row of Q can be computed in time $O(kd)$ as $Z^{(i)} A^{(i)}$ has already been computed in step (1). We finally use $\mathsf{FD}(Q, \alpha, 0)$ to compute a sketch matrix C of Q in time $\tilde{O}(d\alpha^{-3})$. In all, the running time is roughly $O(\mathsf{nnz}(A)k) + \tilde{O}(d\alpha^{-3} + dk\alpha^{-2})$.

3.2. Our Algorithm

Algorithm 6 FFDsparse

Input: $A \in \mathbb{R}^{n\times d}$, $\alpha \in (0,1)$, and integers $k \le d$.

$F = \eta$, $F' = 0$, $B = \mathbf{0}$, $Q = \mathbf{0}$ ▶ η will be determined in Lemma 15

Divide the rows of A into continuous blocks $A^{(1)}, \cdots, A^{(t)}$: we will put new rows into the current block until: a) the number of non-zero entries exceeds dk, or b) the number of rows is $\frac{d}{k}$. When either a) or b) happens, we start a new block. Note that the total number of blocks $t \le \frac{\mathsf{nnz}(A)}{dk} + \frac{nk}{d}$.

Choose subspace embedding matrices J, C (Theorem 6) ▶ J, C are fixed for all blocks

for $i = 1$ **to** t **do**

  Compute $Z^{(i)} = \mathsf{LRA}(A^{(i)}, J, C)$ (Algorithm 5). Let $\ell_i = \mathsf{rows}(A^{(i)})$.

  Compute $A'^{(i)} = Z^{(i)}A^{(i)}$.

  $w = \mathsf{RowNorms}(A^{(i)}, Z^{(i)})$ ▶ w contains the row norms of $A^{(i)} - Z^{(i)T}Z^{(i)}A^{(i)}$ (Algorithm 7)

  $F' = F' + \|w\|^2$, and if $F' \ge 2F$, $F = F'$.

  ▶ We always have $F = O(1)\cdot\sum_i \|(I - Z^{(i)T}Z^{(i)})A^{(i)}\|_F^2$

  Let $p \in \mathbb{R}^{\ell_i}$ such that $p_j = \frac{w_i^2}{\alpha^2 F}$ for $j = 1, \cdots, \ell_i$. Let $x \in \mathbb{R}^{\ell_i}$ be a random vector with independent entries: For each j, $x_j = 1/p_j$ w.p. p_j, and $x_j = 0$ w.p. $1 - p_j$.

  Let $R^{(i)} = (I - Z^{(i)T}Z^{(i)})A^{(i)}$ and $Q^{(i)} = \mathsf{Diag}(x)\cdot R^{(i)}$ (no need to compute $R^{(i)}$ explicitly).

  $B = \mathsf{FFDdense}([B; A'^{(i)}], \alpha, k)$. ▶ Sketching $A' = [A'^{(1)}; \cdots; A'^{(t)}]$ using Theorem 7

  $C = \mathsf{FD}([C; Q^{(i)}], \alpha, 0)$. ▶ Sketching $Q = [Q^{(1)}; \cdots; Q^{(t)}]$ using FD.

end for

return $[B; C]$

Theorem 9 (FFDsparse) *Given any matrix $A \in \mathbb{R}^{n\times d}$, $0 < \alpha < 1$ and $0 \le k \le d$,* $\mathsf{FFDsparse}(A, \alpha, k)$ *(Algorithm 6) maintains a matrix S in a streaming fashion, such that, with probability* 0.9*,*

$$\|A^T A - S^T S\|_2 \le \alpha \|A - [A]_k\|_F^2.$$

The algorithm uses $O(d(k + \alpha^{-1}))$ space and $O(\mathsf{nnz}(A)k + nk^3) + \tilde{O}(d\alpha^{-3} + dk\alpha^{-2})$ time.

By Lemma 3, we also have the following result.

Theorem 10 *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 (ε, k)-*proj*-sketch $S \in \mathbb{R}^{O(k/\varepsilon)\times d}$. The algorithm uses $O(dk/\varepsilon)$ space and runs in $O(\mathsf{nnz}(A)k + nk^3) + \tilde{O}(dk^3\varepsilon^{-3}))$ time.*

3.3. Proof of Theorem 9

The detail of our fast algorithm for sparse matrix is described in Algorithm 6. 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 11 *With probability at least* 0.99*, we have* (1) $\|A' - [A']_k\|_F^2 \le \|A - [A]_k\|_F^2$*;* (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 6, 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.

Let P be the projection matrix onto the subspace spanned by the top-k right singular vectors of A. So we have

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

which proves (1).

As defined in Algorithm 6, $R^{(i)} = (I - Z^{(i)T} Z^{(i)}) A^{(i)}$, where $Z^{(i)} = \mathsf{LRA}(A^{(i)}, J, C)$. Since $\|A^{(i)} - Z^{(i)T} Z^{(i)} A^{(i)}\|_F^2 \le O(1) \cdot \|A^{(i)} - [A^{(i)}]_k\|_F^2$ for each i (by Lemma 8), one gets

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

which proves (2). One caveat: the above argument needs $\|A^{(i)} - Z^{(i)T}Z^{(i)}A^{(i)}\|_F^2 \leq O(1) \cdot \|A^{(i)} - [A^{(i)}]_k\|_F^2$ to hold for all i simultaneously. To achieve this, one could use a similar idea as in Boutsidis et al. (2016) to first boost the success probability in Lemma 8 and then apply a union bound, but this results in an extra log factor in the time complexity.

Thus, we use a different argument to bypass this. Let $S = AC^TJ^T$ and write S in the block form as $S = [S^{(1)}; \cdots; S^{(t)}]$.

Note that $S^{(i)} = A^{(i)}C^TJ^T$ and the rows of $Z^{(i)}$ form an orthonormal basis for the column space of $S^{(i)}$, thus for each i

$$\begin{aligned}
&\min_{X:\mathsf{rank}(X)\leq k} \|A^{(i)} - S^{(i)}X\|_F^2 = \min_{X:\mathsf{rank}(X)\leq k} \|A^{(i)} - Z^{(i)T}X\|_F^2 \\
&= \min_{X:\mathsf{rank}(X)\leq k} \|A^{(i)} - Z^{(i)T}Z^{(i)}A^{(i)} + Z^{(i)T}Z^{(i)}A^{(i)} - Z^{(i)T}X\|_F^2 \\
&= \min_{X:\mathsf{rank}(X)\leq k} \left(\|A^{(i)} - Z^{(i)T}Z^{(i)}A^{(i)}\|_F^2 + \|Z^{(i)T}Z^{(i)}A^{(i)} - Z^{(i)T}X\|_F^2\right) \quad \text{by Pythagorean} \\
&\geq \|A^{(i)} - Z^{(i)T}Z^{(i)}A^{(i)}\|_F^2
\end{aligned}$$

Therefore, we get

$$\begin{aligned}
\min_{X:\mathsf{rank}(X)\leq k} \|A - SX\|_F^2 &= \min_{X:\mathsf{rank}(X)\leq k} \sum_{i=1}^{t} \|A^{(i)} - S^{(i)}X\|_F^2 \\
&\geq \sum_{i=1}^{t} \min_{X:\mathsf{rank}(X)\leq k} \|A^{(i)} - S^{(i)}X\|_F^2. \\
&\geq \sum_{i=1}^{t} \|A^{(i)} - Z^{(i)T}Z^{(i)}A^{(i)}\|_F^2 \qquad (5)
\end{aligned}$$

One the other hand, by Theorem 6, with probability 0.99

$$\min_{X:\mathsf{rank}(X)\leq k} \|A - SX\|_F^2 \leq O(1)\|A - [A]_k\|_F^2. \qquad (6)$$

Combining (5) and (6), we finally get

$$\|R\|_F^2 = \sum_{i=1}^{t} \|R^{(i)}\|_F^2 = \sum_{i=1}^{t} \|A^{(i)} - Z^{(i)T}Z^{(i)}A^{(i)}\|_F^2 \leq O(1)\|A - [A]_k\|_F^2.$$

Note that, in the above proof, we only require one probabilistic event, i.e., (6), to happen, and thus avoid the union bound argument. This finish the proof of part (2). ■

Lemma 12 *$A^TA = R^TR + A'^TA'$; and with probability 0.99, it holds that $\|A^TA - A'^TA'\|_F \leq O(1) \cdot \|A - [A]_k\|_F^2$.*

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

$$\begin{aligned}
R^{(i)T}R^{(i)} &= A^{(i)T}(I - Z^{(i)T}Z^{(i)}) \cdot (I - Z^{(i)T}Z^{(i)})A^{(i)} \\
&= A^{(i)T}(I - Z^{(i)T}Z^{(i)})A^{(i)} && \text{since } (I - Z^{(i)T}Z^{(i)}) \text{ is a projection} \\
&= A^{(i)T}A^{(i)} - A'^{(i)T}A'^{(i)}.
\end{aligned}$$

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 property of the Frobenius norm. Then the second part follows from Lemma 11. ■

Lemma 13 *If the entries of A are integers bounded in magnitude by* $\text{poly}(nd)$ *and* $\mathsf{rank}(A) \ge 1.1k$*, then* $\|A - [A]_k\|_F^2 \ge 1/\text{poly}(nd)$.

Proof The lemma directly follows from a result of Clarkson and Woodruff (2009), and here we use the restated version from Boutsidis et al. (2016).

Lemma 14 (Lemma 37 of Boutsidis et al. (2016)) *If an $n \times d$ matrix A has integer entries bounded in magnitude by γ, and has rank ρ, then the k-th largest singular value of A satisfies*

$$\sigma_k \ge (nd\gamma^2)^{-k/2(\rho-k)}.$$

■

Lemma 15 *We set $\eta = \text{poly}^{-1}(nd)$ (see line 1 of Algorithm 6), then with probability at least* 0.99*, it holds that*

$$\|Q\|_F^2 = O(\|R\|_F^2),\ \|R^TR - Q^TQ\|_2 = O(\alpha)\cdot\|A - [A]_k\|_F^2,$$

$$\textit{and } \mathsf{rows}(Q) = O(\log(nd)/\alpha^2).$$

Proof Let us first assume $\mathsf{rank}(A) \ge 1.1k$. Each row R_i of R is sampled with probability $\Theta(\frac{\|R_i\|^2}{\alpha^2 F})$, with F initialized to be η. We have $\eta \le \|A - [A]_k\|_F^2$ (by Lemma 13). When $\|R\|_F^2 \ge \eta$, the sampling probability for each row is at least $\Omega(\frac{\|R_i\|^2}{\alpha^2\|R\|_F^2})$ (since F is at most a constant approximation to $\|R\|_F^2$ at any time), so the first two parts directly follow from Theorem 5 and Lemma 11. Otherwise if $\|R\|_F^2 \le \eta \le \|A - [A]_k\|_F^2$, then the probability for sampling each row is $\Omega(\frac{\|R_i\|^2}{\alpha^2\eta}) = \Omega(\frac{\|R_i\|^2}{\alpha^2\|A-[A]_k\|_F^2})$ and the first two parts follow from Theorem 5.

To bound the number of rows sampled, we divide the stream into epochs, where F roughly doubles in each epoch. So the total number of epochs is bounded by $O(\log\frac{\|R\|_F}{\eta})$, as the final value of F is at most $O(\|R\|_F^2)$. Recall that we assume each entry of A is an integer bounded in magnitude by $\text{poly}(nd)$, which implies the number of epochs is

$$O(\log\frac{\|R\|_F^2}{\eta}) \le O\left(\log\left(\|A\|_F^2\cdot\text{poly}(nd)\right)\right) = O(\log(nd)).$$

The number of rows sampled in each epoch is at most $O(1/\alpha^2)$: let $a_1, \cdots, a_t$ be the rows of R in the epoch, and thus $\sum_j \|a_j\|^2 \le O(F)$ (since the epoch ends if F doubles); each row

a_j is sampled with probability $\Theta(\frac{\|a_j\|^2}{\alpha^2 F})$, which implies the total number of rows sampled is $O(1/\alpha^2)$. This proves the third part.

The case $\mathsf{rank}(A) \le 1.1k$ 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 11, and thus $\mathsf{rows}(Q) = 0$. In this case, our algorithm is essentially exact. ■

Correctness. By union bound, with probability 0.9, all the above lemmas hold simultaneously, and we assume this happens. Since $C = \mathsf{FD}(Q, \alpha, 0)$, by Theorem 4 and Lemma 15, we have

$$\|Q^T Q - C^T C\|_2 \le \alpha \|Q\|_F^2 \le O(\alpha) \cdot \|R\|_F^2. \tag{7}$$

Since $B = \mathsf{FFDdense}(A', \alpha, k)$, by Theorem 7, we have

$$\|A'^T A' - B^T B\|_2 \le \alpha \|A' - [A']_k\|_F^2 \le \alpha \|A - [A]_k\|_F^2, \tag{8}$$

where the last inequality is from Lemma 11. Let $S = [B; C]$,

$$\begin{aligned}
&\|A^T A - S^T S\|_2 = \|A^T A - B^T B - C^T C\|_2 \\
&\le \|A^T A - A'^T A' - C^T C\|_2 + \|A'^T A' - B^T B\|_2 \\
&\le \|A^T A - A'^T A' - C^T C\|_2 + \alpha \|A - [A]_k\|_F^2 \qquad \text{by (8)} \\
&= \|R^T R - C^T C\|_2 + \alpha \|A - [A]_k\|_F^2 \\
&\le \|R^T R - Q^T Q\|_2 + \|Q^T Q - C^T C\|_2 + \alpha \|A - [A]_k\|_F^2 \quad \text{triangle inequality} \\
&\le O(\alpha) \cdot \|A - [A]_k\|_F^2 + O(\alpha) \cdot \|R\|_F^2 + \alpha \|A - [A]_k\|_F^2 \qquad \text{by (7) and Lemma 15} \\
&\le O(\alpha) \cdot \|A - [A]_k\|_F^2, \qquad \text{by Lemma 11}
\end{aligned}$$

which proves the error bound after adjusting α by a constant.

Running time. Let ℓ_i be the number of rows in ith block $A^{(i)}$. The time to compute $Z^{(i)}$ using Lemma 8 is $O(\mathsf{nnz}(A^{(i)}) + \ell_i k^2)$. Hence the total time used on this step is

$$\sum_i O(\mathsf{nnz}(A^{(i)}) + \ell_i k^2) = O(\mathsf{nnz}(A) + nk^2).$$

The step to compute the matrix multiplication $A'^{(i)} = Z^{(i)} A^{(i)}$ takes $O(\mathsf{nnz}(A^{(i)})k)$ time, since Z has $O(k)$ rows. So the total time spent on this step is $O(\mathsf{nnz}(A)k)$. By definition, there are at most $O(\frac{\mathsf{nnz}(A)}{dk} + \frac{nk}{d})$ blocks. Moreover, after left multiplied by $Z^{(i)}$, each block contributes $O(k)$ rows to A', and thus the total number of rows in A' is at most $O(\frac{\mathsf{nnz}(A)}{d} + \frac{nk^2}{d})$. Computing B by invoking $B = \mathsf{FFDdense}(A', \alpha, k)$ needs $O(\mathsf{rows}(A')dk) + \tilde{O}(d/\alpha^3) = O(\mathsf{nnz}(A)k + nk^3) + \tilde{O}(d/\alpha^3)$ time. Finally, each row of Q can be computed in time $O(dk)$ given A' (which has been computed in line 6). Invoking $C = \mathsf{FD}(Q, \alpha, 0)$ needs $\tilde{O}(d/\alpha^3 + dk/\alpha^2)$ since $\mathsf{rows}(Q) = \tilde{O}(1/\alpha^2)$ by Lemma 15. So far, the total time is $O(\mathsf{nnz}(A)k + nk^3) + \tilde{O}(d\alpha^{-3} + dk\alpha^{-2})$.

The time for naively computing the row norms of R is high. One may use Johnson-Lindenstrauss transforms (Johnson and Lindenstrauss, 1984) to reduce the running time,

since constant approximations are good enough. By standard arguments, it is not hard to verify that the running time will be $O(\mathsf{nnz}(A)\log n)$ with JL transforms. If $k = \Omega(\log n)$, this term is absorbed by the leading term $O(\mathsf{nnz}(A)k)$; if $k = o(\log n)$, however, this becomes the dominating part. We next provide an algorithm, which computes the *exact* row norms of $A - Z^T ZA$ in time $O(\mathsf{nnz}(A)k)$. Using this algorithm, the total time of FFDsparse is $O(\mathsf{nnz}(A)k + nk^3) + \tilde{O}(d\alpha^{-3} + dk\alpha^{-2})$.

Faster algorithm for computing row norms.

Lemma 16 *Given any $A \in \mathbb{R}^{n\times d}$ and $Z \in \mathbb{R}^{O(k)\times n}$, the exact row norms of $A - Z^T ZA$ can be computed in time $O(\mathsf{nnz}(A)k + dk^2 + nk^2)$ and in space $O(nk + dk)$.*

Proof The algorithm is presented in Algorithm 7. We first compute ZA in $\mathsf{nnz}(A)k$ time,

Algorithm 7 RowNorms

Input: $A \in \mathbb{R}^{n\times d}$, $Z \in \mathbb{R}^{O(k)\times \ell}$

1: Compute $S = ZA$
2: Compute $[U, \Sigma, V] = \mathsf{SVD}(S)$
3: Compute $L = Z^T U\Sigma$
4: **for** $i = 1$ **to** n **do**
5: Compute $\mathbf{a}^{(1)} = A_i V$
6: Compute $w_i = \left\|\mathbf{a}^{(1)} - L_i\right\|^2 + \|A_i\|^2 - \|\mathbf{a}^{(1)}\|^2$
7: **end for**
8: **return** $w = [w_1, \cdots, w_n]$

then perform SVD on ZA: $ZA = U\Sigma V^T$. Since ZA is a $O(k) \times d$ matrix, this takes $O(dk^2)$ time. Next compute $L = Z^T U\Sigma$, which takes $O(\mathsf{rows}(A)k^2)$ time. Let $\mathbf{a}$ be the ith row in A. We compute $\mathbf{a}^{(1)} = \mathbf{a}V$ (in $O(\mathsf{nnz}(a)k)$ time), and set $\mathbf{b} = \mathbf{a} - \mathbf{a}^{(1)}V^T$ (we don't compute $\mathbf{b}$ explicitly). Then decompose $\mathbf{a}$ into two orthogonal parts as

$$\mathbf{a} = \mathbf{a}^{(1)}V^T + \mathbf{b}.$$

The ith row of $A - Z^T ZA$ is

$$\mathbf{a} - L_i V^T = \left(\mathbf{a}^{(1)} - L_i\right) V^T + \mathbf{b}.$$

Note that V^T and $\mathbf{b}$ are orthogonal, and thus

$$\|\mathbf{a} - L_i V^T\|^2 = \|\left(\mathbf{a}^{(1)} - L_i\right) V^T\|^2 + \|\mathbf{b}\|^2 = \left\|\mathbf{a}^{(1)} - L_i\right\|^2 + \|\mathbf{a}\|^2 - \|\mathbf{a}^{(1)}\|^2,$$

which can be computed in time $O(\mathsf{nnz}(\mathbf{a}) + k)$. In all, the total time to compute all the row norms in $A - Z^T ZA$ is

$$\mathsf{nnz}(A)k + dk^2 + \mathsf{rows}(A)k^2 + \sum_{i=1}^{\mathsf{rows}(A)} O(\mathsf{nnz}(A_i) + k) = O(\mathsf{nnz}(A)k + dk^2 + \mathsf{rows}(A)k^2).$$

■

Therefore, the time to compute all the row norms in the jth block $A^{(j)} - Z^T Z A^{(j)}$ is $O(\mathsf{nnz}(A^{(j)})k + dk^2 + \mathsf{rows}(A^{(j)})k^2)$, and the total time over all block is thus

$$\sum_{j=1}^{t} O(\mathsf{nnz}(A^{(i)})k + dk^2 + \mathsf{rows}(A^{(i)})k^2) = O(\mathsf{nnz}(A)k + nk^3),$$

where we use the fact that $t = O(\frac{\mathsf{nnz}(A)}{dk} + \frac{nk}{d})$.

Space. For space, we need a buffer to store a new block of A, the size of which is at most $dk + d$, as $\mathsf{nnz}(A^{(i)})$ is at most $dk + d$. When using Algorithm 5, the input matrix always has at most $\frac{d}{k}$ rows, so we need $O(dk)$ space to compute and store each $Z^{(i)}$ (Lemma 8). $A'^{(i)}$ is of dimension $O(k) \times d$, which needs $O(dk)$ space to compute and store. According to Lemma 16, the space needed to compute the row norms of each $R^{(i)}$ is $O(\ell_i k + dk) = O(dk)$, since $\ell_i \leq \frac{d}{k}$ for all i. From Theorem 7, the space used by $\mathsf{FFDdense}(A', \alpha, k)$ is $O(d(k+\alpha^{-1}))$. Note that, in line 12 of Algorithm 6, the rows of $Q^{(i)}$ can be computed one by one and fed to FD directly, and thus compute $C = \mathsf{FD}(Q, \alpha, 0)$ uses $O(d/\alpha)$ space by theorem 4. In all, the total space usage is bounded by $O(d(k + \alpha^{-1}))$.

4. On the Equivalence Between Frequent Directions and Matrix Multiplication

In this section, we show that, under mild assumptions, the time complexity of Frequent Directions is essentially equivalent to that of matrix multiplication modulo lower order terms.

Let $\mathcal{A}$ be an algorithm for multiplying a matrix $A \in \mathbb{R}^{n \times d}$ with an arbitrary k by n matrix; and $T(\mathsf{nnz}(A), k, n, d)$ be the running time. In this section we will assume T is non-decreasing in the first parameter $\mathsf{nnz}(A)$. Moreover, T is additive in the first parameter, i.e., $T(a, k, n, d) + T(b, k, n, d) = T(a + b, k, n, d)$ for any non-negative a and b.

4.1. Faster Matrix Multiplication Implies Faster FD

This direction follows from our FFDsparse algorithm.

Theorem 17 *Let $\mathcal{A}$ be an algorithm for multiplying a matrix $A \in \mathbb{R}^{n \times d}$ with an arbitrary k by n matrix and assume its running time is $T(\mathsf{nnz}(A), k, n, d)$. If T is additive and non-decreasing in the first parameter $\mathsf{nnz}(A)$, then Algorithm 6 can be implemented in time $T(\mathsf{nnz}(A), O(k), \frac{d}{k}, d) + O(nk^3) + \tilde{O}(d\alpha^{-3} + dk\alpha^{-2})$.*

Proof First, we note that the time complexity of our algorithm for sketching sparse matrices is dominated by computing $Z^{(i)}A^{(i)}$ (Line 6 in Algorithm 6) for $i = 1, 2, \cdots, t$, where $Z^{(i)}$ contains $O(k)$ rows. Using $\mathcal{A}$, this takes $T(\mathsf{nnz}(A^{(i)}), O(k), \frac{d}{k}, d)$ time, since the number of rows in $A^{(i)}$ is at most d/k in the algorithm. By the additive assumption, the total time is

$$\sum_{i=1}^{t} T(\mathsf{nnz}(A^{(i)}), O(k), \frac{d}{k}, d) = T(\mathsf{nnz}(A), O(k), \frac{d}{k}, d).$$

■

4.2. Faster FD Implies Faster Matrix Multiplication

In this subsection, we prove the other direction by showing that the running time of FD is lower bounded by matrix multiplication. The proof is based on the idea of Musco and Woodruff (2017). In particular, we show that the existence an algorithms which computes an $(O(1), k)$-cov-sketch in time $o(\mathsf{nnz}(A)k)$ 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 18 *Assume there is an algorithm $\mathcal{A}$, which, given any $A \in \mathbb{R}^{n\times d}$ with polynomially bounded integer entries, returns $B \in \mathbb{R}^{O(k)\times d}$ in time $T(\mathsf{nnz}(A), k, n, d)$ such that*

$$\|A^TA - B^TB\|_2 \leq \Delta\|A - [A]_k\|_F^2,$$

for some constant error parameter Δ. If T is additive and non-decreasing in the first parameter, then there is a $T(\mathsf{nnz}(M), k, n, d) + T(nk, k, n, d) + O(dk^2)$ time algorithm for multiplying arbitrary polynomially bounded integer matrices $M^T \in \mathbb{R}^{(d-k)\times n}$, $C \in \mathbb{R}^{n\times k}$.

Proof For any matrices $M \in \mathbb{R}^{n\times(d-k)}$ and $C \in \mathbb{R}^{n\times k}$ with integer entries in $[-U, U]$, let $A \in \mathbb{R}^{n\times d}$ 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 $\|A - [A]_k\|_F^2 \leq \|M\|_F^2 \leq ndU^2$ and

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

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

$$\|A^TA - B^TB\|_2 \leq \Delta\|A - [A]_k\|_F^2 \leq \Delta U^2nd,$$

for some constant error parameter Δ.

The spectral norm of a matrix N is the largest singular value, which can be equivalently defined as $\|N\|_2 = \max_{x,y:\|x\|=\|y\|=1} x^TNy$, thereby $N_{i,j} = e_i^TNe_j \leq \|N\|_2$ for all i, j. It follows that $(A^TA - B^TB)_{i,j} \leq \Delta U^2nd$, meaning the corresponding block of B^TB is an entry-wise approximation to wM^TC within additive error ΔU^2nd.

Now if w is a large integer, say $w = \lceil 3\Delta U^2nd\rceil$, we can recover M^TC from B^TB exactly by rounding the numbers in B^TB to their nearest integer multiple of w (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 $\mathcal{A}$, the exact integer matrix multiplication M^TC can be computed in time

$$T(\mathsf{nnz}(A), k, n, d) + O(dk^2) = T(\mathsf{nnz}(M), k, n, d) + T(nk, k, n, d) + O(dk^2).$$

Here we used the fact that $\mathsf{nnz}(wC) \leq nk$ and the assumption that T is linear in the first parameter. 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. ■

Therefore, if $T = o(\mathsf{nnz}(A)k) = o(\mathsf{nnz}(M)k + nk^2)$, then M^TC can be computed in time $o(\mathsf{nnz}(M)k) + O(nk^2 + dk^2)$, which will be a breakthrough in fast matrix multiplication.

5. Conclusion

This paper studies the problem of computing covariance sketches for matrices in the streaming model. Based on several novel algorithmic techniques, we provide new space-optimal algorithms with improved running time. We also prove that, under mild assumptions, the time complexity of Frequent Directions is essentially equivalent to that of matrix multiplication up to lower order terms. In particular, this implies that the running times of FFDdense and FFDsparse cannot be significantly improved unless the state-of-the-art matrix multiplication algorithms can, and vice versa. Thus, this paper almost settles the time complexity of this problem.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No. 61802069), Shanghai Sailing Program (Grant No. 18YF1401200) and Shanghai Science and Technology Commission (Grant No. 17JC1420200).

Appendix A. Omitted Proofs

A.1. Covariance error to projection error

Lemma 19 $\|A - \pi_B^k(A)\|_F^2 \leq \|A - [A]_k\|_F^2 + 2k \cdot \|A^T A - B^T B\|_2$.

Proof For any x with $\|x\| = 1$, we have

$$\left| \|Ax\|^2 - \|Bx\|^2 \right| = \left| x^T (A^T A - B^T B) x \right| \leq \|A^T A - B^T B\|_2 \tag{9}$$

Let u_i and w_i be the ith right singular vector of B and A respectively

$$\begin{aligned}
&\|A - \pi_B^k(A)\|_F^2 = \|A\|_F^2 - \|\pi_B^k(A)\|_F^2 \\
&= \|A\|_F^2 - \sum_{i=1}^{k} \|Au_i\|^2 \quad \text{Pathagorean theorem} \\
&\leq \|A\|_F^2 - \sum_{i=1}^{k} \|Bu_i\|^2 + k \cdot \|A^T A - B^T B\|_2 \quad \text{by Eq. (9)} \\
&\leq \|A\|_F^2 - \sum_{i=1}^{k} \|Bw_i\|^2 + k \cdot \|A^T A - B^T B\|_2 \quad \text{because } \sum_{i=1}^{k} \|Bw_i\|^2 \leq \sum_{i=1}^{k} \|Bu_i\|^2 \\
&\leq \|A\|_F^2 - \sum_{i=1}^{k} \|Aw_i\|^2 + 2k \cdot \|A^T A - B^T B\|_2 \quad \text{by Eq. (9)} \\
&= \|A - [A]_k\|_F^2 + 2k \cdot \|A^T A - B^T B\|_2.
\end{aligned}$$

■

A.2. Row sampling

Theorem 20 *For any $A \in \mathbb{R}^{n \times d}$ and $F > 0$, we sample each row A_i with probability $p_i \geq \frac{\|A_i\|^2}{\alpha^2 F}$; if it is sampled, scale it by $1/\sqrt{p_i}$. Let B be the (rescaled) sampled rows, then w.p. 0.99, $\|A^T A - B^T B\|_2 \leq 10\alpha\sqrt{F}\|A\|_F$, and $\|B\|_F \leq 10\|A\|_F$. The expected number of rows sampled is $O(\frac{\|A\|_F^2}{\alpha^2 F})$ if $p_i = O(\frac{\|A_i\|^2}{\alpha^2 F})$ for each i.*

Proof Since spectral norm is no larger than the Frobenius norm, it is sufficient to prove $\|A^T A - B^T B\|_F \leq 10\alpha\sqrt{F}\|A\|_F$.

For each $j \in [n]$, let

$$x_j = \begin{cases} 1 & \text{if the } j\text{th rows of } A \text{ is sampled} \\ 0 & \text{otherwise.} \end{cases}$$

We have $(A^T A)_{i,j} = \sum_{t=1}^{n} a_{t,i} a_{t,j}$, while

$$(B^T B)_{i,j} = \sum_{t=1}^{n} \frac{x_t^2 \cdot a_{t,i} a_{t,j}}{p_t}.$$

So $\mathsf{E}[(B^TB)_{i,j}] = (A^TA)_{i,j}$. We also have

$$\mathsf{Var}[(B^TB)_{i,j}] = \mathsf{Var}\left[\sum_{t=1}^{n} \frac{x_t^2 \cdot a_{t,i}a_{t,j}}{p_t}\right] = \sum_{t=1}^{n} \frac{a_{t,i}^2 a_{t,j}^2 \cdot \mathsf{Var}\left[x_t^2\right]}{p_t^2} \leq \sum_{t=1}^{n} \frac{a_{t,i}^2 a_{t,j}^2}{p_t},$$

where we use the fact $\mathsf{Var}[x_t^2] = p_t(1-p_t) \leq p_t$. So we have

$$\mathsf{E}\left[\left((A^TA)_{i,j} - (B^TB)_{i,j}\right)^2\right] = \mathsf{Var}[(B^TB)_{i,j}] \leq \sum_{t=1}^{n} \frac{a_{t,i}^2 a_{t,j}^2}{p_t}.$$

Therefore,

$$\begin{aligned}
\mathsf{E}\left[\|A^TA - B^TB\|_F^2\right] &= \sum_{i,j} \mathsf{E}\left[\left((A^TA)_{i,j} - (B^TB)_{i,j}\right)^2\right] \\
&\leq \sum_{i,j}\sum_{t=1}^{n} \frac{a_{t,i}^2 a_{t,j}^2}{p_t} \\
&= \sum_{t=1}^{n} \frac{\|A_t\|^2\|A_t\|^2}{p_t} \\
&= \sum_{t=1}^{n} \alpha^2 F\|A_t\|^2 = \alpha^2 F\|A\|_F^2.
\end{aligned} \tag{10}$$

We adjust α by a constant, and using Markov's inequality

$$\mathsf{Pr}\left[\|A^TA - B^TB\|_F^2 \geq 100\alpha^2 F\|A\|_F^2\right] \leq 0.01,$$

which is equivalent to

$$\mathsf{Pr}\left[\|A^TA - B^TB\|_F \geq 10\alpha\sqrt{F}\|A\|_F\right] \leq 0.01.$$

The success probability can be boosted by a similar argument as in Drineas et al. (2006) via McDiarmid's inequality (see e.g. Boucheron et al. (2013)).

It is not hard to verify that

$$\mathsf{E}\left[\|B\|_F^2\right] = \|A\|_F^2.$$

So by another Markov inequality, we prove the second part. ■

References

Haim Avron, Vikas Sindhwani, and David Woodruff. Sketching structured matrices for faster nonlinear regression. In *NIPS*, 2013.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

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

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

Daniele Calandriello, Alessandro Lazaric, and Michal Valko. Efficient second-order online kernel learning with adaptive embedding. In *NIPS*, 2017.

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

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

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

Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *STOC*. ACM, 2015.

Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *SODA*. SIAM, 2017.

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

Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *SODA*. SIAM, 2013.

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

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

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

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

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

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

Ravi Kannan, Santosh Vempala, and David Woodruff. Principal component analysis and higher correlations for distributed data. In *COLT*, 2014.

Zohar Karnin and Edo Liberty. Online pca with spectral bounds. In *COLT*, 2015.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. Scalable semi-supervised query classification using matrix sketching. In *ACL*, 2016.

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

Yingyu Liang, Maria-Florina F Balcan, Vandana Kanchanapally, and David Woodruff. Improved distributed principal component analysis. In *NIPS*, 2014.

Edo Liberty. Simple and deterministic matrix sketching. In *KDD*. ACM, 2013.

Haipeng Luo, Alekh Agarwal, Nicolo Cesa-Bianchi, and John Langford. Efficient second order online learning by sketching. In *NIPS*, 2016.

Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *NIPS*, 2015.

Cameron Musco and David Woodruff. Is input sparsity time possible for kernel low-rank approximation? In *NIPS*, 2017.

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

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

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

David Woodruff. Low rank approximation lower bounds in row-update streams. In *NIPS*, 2014a.

David Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014b.

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

Haida Zhang, Zengfeng Huang, Zhewei Wei, Wenjie Zhang, and Xuemin Lin. Tracking matrix approximation over distributed sliding windows. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 2017.

Yuchen Zhang, Martin Wainwright, and Michael Jordan. Distributed estimation of generalized matrix rank: Efficient algorithms and lower bounds. In *ICML*, 2015.