Optimal Principal Component Analysis in Distributed and Streaming Models

Christos Boutsidis
Goldman Sachs
New York, USA
christos.boutsidis@gmail.com

David P. Woodruff
IBM Research
California, USA
dpwoodru@us.ibm.com

Peilin Zhong
Institute for Interdisciplinary Information Sciences
Tsinghua University, China
zpl12@mails.tsinghua.edu.cn

ABSTRACT

This paper studies the Principal Component Analysis (PCA) problem in the distributed and streaming models of computation. Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, a rank parameter $k < \text{rank}(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$, we want to output an $m \times k$ orthonormal matrix $\mathbf{U}$ for which

$$\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2,$$

where $\mathbf{A}_k \in \mathbb{R}^{m \times n}$ is the best rank-k approximation to $\mathbf{A}$.

Our contributions are summarized as follows:

1. In the arbitrary partition distributed model of Kannan et al. (COLT 2014), each of s machines holds a matrix $\mathbf{A}^i$ and $\mathbf{A} = \sum_{i=1}^s \mathbf{A}^i$. Each machine should output $\mathbf{U}$. Kannan et al. achieve $O(skm/\varepsilon) + \text{poly}(sk/\varepsilon)$ words (of $O(\log(nm))$ bits) communication. We obtain the improved bound of $O(skm) + \text{poly}(sk/\varepsilon)$ words, and show an optimal (up to low order terms) $\Omega(skm)$ lower bound. This resolves an open question in the literature. A $\text{poly}(\varepsilon^{-1})$ dependence is known to be required, but we separate this dependence from m.

2. In a more specific distributed model where each server receives a subset of columns of $\mathbf{A}$, we bypass the above lower bound when $\mathbf{A}$ is *ϕ-sparse* in each column. Here we obtain an $O(sk\phi/\varepsilon) + \text{poly}(sk/\varepsilon)$ word protocol. Our communication is independent of the matrix dimensions, and achieves the guarantee that each server, in addition to outputting $\mathbf{U}$, outputs a subset of $O(k/\varepsilon)$ columns of $\mathbf{A}$ containing a $\mathbf{U}$ in its span (that is, for the first time, we solve distributed column subset selection). Additionally, we show a matching $\Omega(sk\phi/\varepsilon)$ lower bound for distributed column subset selection. Achieving our communication bound when $\mathbf{A}$ is sparse in general but not sparse in each column, is impossible.

3. In the streaming model of computation, in which the columns of the matrix $\mathbf{A}$ arrive one at a time, an algorithm of Liberty (KDD, 2013) with an improved analysis by Ghashami and Phillips (SODA, 2014) achieves $O(km/\varepsilon)$ "real numbers" space complexity. We improve this result, since our one-pass streaming PCA algorithm achieves an $O(km/\varepsilon) + \text{poly}(k/\varepsilon)$ word space upper bound. This almost matches a known $\Omega(km/\varepsilon)$ bit lower bound of Woodruff (NIPS, 2014). We show that with two passes over the columns of $\mathbf{A}$ one can achieve an $O(km) + \text{poly}(k/\varepsilon)$ word space upper bound; another lower bound of Woodruff (NIPS, 2014) shows that this is optimal for any constant number of passes (up to the $\text{poly}(k/\varepsilon)$ term and the distinction between words versus bits).

4. Finally, in turnstile streams, in which we receive entries of $\mathbf{A}$ one at a time in an arbitrary order, we describe an algorithm with $O((m+n)k\varepsilon^{-1})$ words of space. This improves the $O((m+n)\varepsilon^{-2})k\varepsilon^{-2})$ upper bound of Clarkson and Woodruff (STOC 2009), and matches their $\Omega((m+n)k\varepsilon^{-1})$ word lower bound.

Notably, our results do not depend on the condition number or any singular value gaps of $\mathbf{A}$.

Categories and Subject Descriptors

G.1.3 [**Mathematics of Computing**]: Numerical Analysis - Numerical Linear Algebra; E.m [**Data**]: Miscellaneous

General Terms

Algorithms, Theory

Keywords

low rank matrix decomposition, singular value decomposition, principal component analysis, column subset selection, distributed, streaming, sparse, lower bounds

1. INTRODUCTION

In distributed-memory computing systems such as, for example, Hadoop [1] or Spark [3], Principal Component Analysis (PCA) and the related Singular Value Decomposition (SVD) of large matrices is becoming very challenging. Machine learning libraries implemented on top of such systems, for example mahout [2] or mllib [4], provide distributed PCA implementations since PCA is often used as a building block for a learning algorithm. PCA is useful for dimension reduction, noise removal, visualization, etc. In all of these implementations, the bottleneck is the communication; hence, the focus has been on minimizing the communication cost of the related algorithms, and not the computational cost, which is often the bottleneck in more traditional batch systems.

The data matrix corresponding to the dataset in hand, e.g., a term-document matrix representing a text collection, or the Netflix matrix representing user's ratings for different movies, could be distributed in many different ways [38]. In this paper, we focus on the following so-called arbitrary partition and column partition models. In the arbitrary partition

STOC'16, June 19–21, 2016, Cambridge, MA, USA
© 2016 ACM. 978-1-4503-4132-5/16/06...$15.00
http://dx.doi.org/10.1145/2897518.2897646

model, a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is arbitrarily distributed among s machines. Specifically, $\mathbf{A} = \sum_{i=1}^{s} \mathbf{A}_i$ where $\mathbf{A}_i \in \mathbb{R}^{m\times n}$ is held by machine i. Unless otherwise stated, we always assume $m \leq n$. In the column partition model, a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is distributed column-wise among $s < n$ machines: $\mathbf{A} = \begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2 & \dots & \mathbf{A}_s\end{pmatrix}$; here, for $i = 1 : s$, $\mathbf{A}_i$ is an $m \times w_i$ column submatrix of $\mathbf{A}$ with $\sum_i w_i = n$. Note that the column partition model is a special case of the arbitrary partition model. Thus, it is desirable to prove upper bounds in the arbitrary partition model, and lower bounds in the column partition model. Both models have been adapted by traditional numerical linear algebra software libraries for distributed memory matrix computations [8].

A recent body of work [24, 6, 33, 26, 30, 9] (see Section 2 for a comparison) has focused on designing algorithms which minimize the communication needed for each machine to output an $m \times k$ matrix $\mathbf{U}$ for which $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\cdot\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$, where $\mathbf{A}_k \in \mathbb{R}^{m\times n}$ is the best rank-k approximation to $\mathbf{A}$. Each machine should output the same matrix $\mathbf{U}$ which can then be used for downstream applications, such as clustering, where one first projects the data to a lower dimensional subspace (see, e.g., [6]). The protocol should succeed with large constant probability. The model is such that each of the s machines communicates with one machine, called the "coordinator" (which we sometimes refer to as Server), but does not communicate with other machines. This is known as the coordinator model and one can simulate arbitrary point-to-point communication with a constant factor overhead in communication together with an additive $O(\log(mn))$ bit overhead per message (see, e.g., [37]).

1.1 Our Results

1.1.1 Distributed Models

In the arbitrary partition distributed model, the best upper bound is $O((skm\varepsilon^{-1}) + s\cdot\mathrm{poly}(k\varepsilon^{-1}))$ words (of $O(\log(mn))$ bits) communication [30], while the only known lower bound for this problem is in the arbitrary partition model and is $\Omega(skm)$ [30]. In high precision applications one may want to set ε to be as small as possible and the leading order term of $O(skm\varepsilon^{-1})$ is undesirable. A natural question is whether there is an $O((skm) + s\cdot\mathrm{poly}(k\varepsilon^{-1}))$ word protocol. We note that there is a fairly simple known $\Omega(\varepsilon^{-2})$ bit lower bound [46], so although one cannot achieve an $O(\log(1/\varepsilon))$ dependence in the communication as one would maybe expect given iterative algorithms for regression with running time $O(\log(1/\varepsilon))$ (see section 2.6 of [47] for an overview), an $O((skm) + s\cdot\mathrm{poly}(k\varepsilon^{-1}))$ bound would at least separate the dependence of ε and m and allow for much smaller ε with the same amount of communication.

In the column-partition distributed model, there are many existing protocols using $O(skm\varepsilon^{-1})$ words of communication [24, 6, 33, 26, 30, 9], and the protocols work in very different ways: that of [24, 6] is based on coresets, while that of [33, 26] is based on adapting a streaming algorithm to the distributed setting, while that of [30] is based on sketching, and that of [9] is based on alternating minimization and is useful only under some assumptions on the condition number (see Section 2 for a more detailed discussion of these protocols). It was thus natural to assume there should be a lower bound of $\Omega(skm\varepsilon^{-1})$. Instead, we obtain a new upper bound in the arbitrary partition model and a matching lower bound (up to lower order terms) in the column partition model.

THEOREM 1. *(Restatement of Theorems 8 and 15) Suppose the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is partitioned in the arbitrary-partition model (See Definition 1). For any $1 \geq \varepsilon > 0$, there is an algorithm which on termination leaves the same orthonormal matrix $\mathbf{U} \in \mathbb{R}^{m\times k}$ on each machine such that $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\cdot\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$ holds with arbitrarily large constant probability. Further, the algorithm runs in polynomial time with total communication complexity $O(skm + s\cdot poly(k\varepsilon^{-1}))$ words each containing $O(\log(smn\varepsilon^{-1}))$ bits.*

Suppose the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is partitioned in the column-partition model (See Definition 2). For any positive $1 \leq C \leq O(\mathrm{poly}(skm))$, any algorithm which on termination leaves the same orthonormal matrix $\mathbf{U} \in \mathbb{R}^{m\times k}$ on each machine for which $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq C\cdot\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$ holds with constant probability, requires $\Omega(skm\log(skm))$ bits of communication.

In some applications even an $O(skm)$ word protocol may be too costly, as m could be very large. One could hope to do for communication what recent work on input sparsity algorithms [18, 35, 36, 12, 21] has done for computation, i.e., obtain protocols sensitive to the number of non-zero entries of $\mathbf{A}$. Many matrices are sparse, e.g., Netflix provides a training data set of $100,480,507$ ratings that $480,189$ users give to $17,770$ movies. If users correspond to columns in the matrix, then the average column sparsity is ≈ 200 non-zero elements (out of the $17,770$ possible coordinates).

Our second contribution is a distributed PCA algorithm in the column-partition model with communication complexity depending on the number of non-zero entries of $\mathbf{A}$. The communication complexity of this algorithm is independent of the matrix dimensions. Denote by ϕ the maximum number of non-zero elements of a column in $\mathbf{A}$. When we say that $\mathbf{A}$ is ϕ-sparse, we mean that every column of $\mathbf{A}$ has at most ϕ non-zero elements and $\mathrm{nnz}(\mathbf{A}) \leq \phi\cdot n$. Our protocol has the additional feature of leaving on each machine the same subset $\mathbf{C}$ of $O(k/\varepsilon)$ columns of $\mathbf{A}$ for which there exists an $m\times k$ orthonormal matrix $\mathbf{U}$ in the column span of $\mathbf{C}$ for which $\|\mathbf{A} - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}\|_{\mathrm{F}}^2 \leq \|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\cdot\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$. This is known as the *column subset selection problem*, which is useful since $\mathbf{C}$ may be sparse if $\mathbf{A}$ is, and also it can lead to better data intepretability. To partially complement our protocol, we show our protocol is optimal for any protocol solving the column subset selection problem.

THEOREM 2. *(Restatement of Theorem 12) Suppose a ϕ-sparse matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is partitioned in the column-partition model (See Definition 2). For any $1 \geq \varepsilon > 0$, there is an algorithm which on termination leaves $\mathbf{C} \in \mathbb{R}^{m\times c}$ with $c = O(k/\varepsilon)$ columns of $\mathbf{A}$ and an orthonormal matrix $\mathbf{U}$ on each machine such that $\|\mathbf{A} - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}\|_{\mathrm{F}}^2 \leq \|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\cdot\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$ holds with arbitrarily large constant probability. Further, the algorithm runs in polynomial time with total communication complexity $O\left(sk\phi\varepsilon^{-1} + sk^2\varepsilon^{-4}\right)$ words, each of which is specified by $O(\log(smn\varepsilon^{-1}))$ bits.*

1.1.2 Streaming Models

A model closely related to the distributed model of computation is the streaming model of computation. The model we focus on is the so-called *turnstile streaming model*. In this model, there is a *stream* of update operations and each operation indicates that the corresponding entry of $\mathbf{A}$ should be incremented by a specific number.

Our first one-pass algorithm improves upon the best existing streaming PCA algorithm [33, 26] in two respects. First,

	Upper bounds		Lower bounds	
Definition 3 (arbitrary partition model)	$O(skm + s \cdot \text{poly}(k/\varepsilon))$	(Theorem 8)	$\Omega(skm)$	(Theorem 1.2 in [30])
Definition 4 (column partition model)	$O(skm + s \cdot \text{poly}(k/\varepsilon))$	(Theorem 8)	$\Omega(skm)$	(Theorem 15)
Definition 4 with sparsity $\phi = o(\varepsilon \cdot m)$	$O(sk\phi\varepsilon^{-1} + s \cdot \text{poly}(k/\varepsilon))$	(Theorem 12)	$\Omega(sk\phi)$	(Corollary 2)

Table 1: **Communication upper/lower bounds for the Distributed PCA problems.**

the space of [33, 26] is described in "real numbers" while our space bound ($O(mk/\varepsilon + \text{poly}(k/\varepsilon))$ - see Theorem 9) is in terms of words (we also bound the word size). This matches an $\Omega(km/\varepsilon)$ bit lower bound for one-pass algorithms in [45], up to the distinction between words versus bits and a low order term $\text{poly}(k/\varepsilon)$. Second, our algorithm can be applied in the turnstile streaming model which is stronger than the column update streaming model in [33, 26].

THEOREM 3. *(Restatement of Theorem 9) Suppose a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is given by a stream of update operations in the turnstile streaming model (See Definition 7). For any $1 \geq \varepsilon > 0$, there is an algorithm which uses a single pass over the stream and on termination outputs an orthonormal matrix $\mathbf{U} \in \mathbb{R}^{m\times k}$ such that $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$ holds with arbitrarily large constant probability. Further, the algorithm runs in polynomial time with space of total $O\left(mk/\varepsilon + \text{poly}(k\varepsilon^{-1})\right)$ words each containing $O(\log(smn\varepsilon^{-1}))$ bits.*

A slight modification of the previous algorithm leads to a one-pass algorithm which can compute a factorization of a $(1+\varepsilon)$-approximate rank-k matrix. The modified algorithm only needs $O((n+m)k/\varepsilon + \text{poly}(k/\varepsilon))$ words of space which improves the $O((m + n\varepsilon^{-2})k\varepsilon^{-2})$ upper bound in [17] and matches the $\Omega((n+m)k/\varepsilon)$ bit lower bound given by [17], up to the low order term $\text{poly}(k/\varepsilon)$.

THEOREM 4. *(Restatement of Theorem 10) Suppose a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is given by a stream of update operations in the turnstile streaming model (See Definition 7). For any $1 \geq \varepsilon > 0$, there is an algorithm which uses a single pass over the stream and on termination outputs a factorization of a matrix $\mathbf{A}_k^* \in \mathbb{R}^{m\times n}$ with $rank(\mathbf{A}_k^*) \leq k$ such that $\|\mathbf{A} - \mathbf{A}_k^*\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$ holds with arbitrarily large constant probability. Further, the algorithm runs in polynomial time with space of total $O((m+n)k/\varepsilon + \text{poly}(k\varepsilon^{-1}))$ words each containing $O(\log(smn\varepsilon^{-1}))$ bits.*

We also show a two-pass algorithm which is an implementation of our distributed PCA protocol. It uses $O((km) + \text{poly}(k/\varepsilon))$ words of space, which up to the $\text{poly}(k/\varepsilon)$ term and the distinction between words versus bits, is optimal for any constant number of passes. A "next natural goal" in [45] was to improve the lower bound of $\Omega(km)$ to a bound closer to the 1-pass $\Omega(km/\varepsilon)$ lower bound established in that paper; our upper bound shows this is not possible.

THEOREM 5. *(Restatement of Theorem 11) Suppose a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is given by a stream of update operations in the turnstile streaming model (See Definition 7). For any $1 \geq \varepsilon > 0$, there is an algorithm which uses two passes over the stream and upon termination outputs an orthonormal matrix $\mathbf{U} \in \mathbb{R}^{m\times k}$ such that $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$ holds with arbitrarily large constant probability. Further, the algorithm runs in polynomial time with total space $O\left(mk + \text{poly}(k\varepsilon^{-1})\right)$ words each containing $O(\log(smn\varepsilon^{-1}))$ bits.*

1.2 Technical Overview

1.2.1 Upper Bounds

The Arbitrary Partition Model.

Recall that the input matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is arbitrarily partitioned into s matrices $\mathbf{A}_i \in \mathbb{R}^{m\times n}$, i.e., for $i = 1, 2, \ldots, s$: $\mathbf{A} = \sum_{i=1}^{s} \mathbf{A}_i$. Critical to our protocol is the recent notion of *projection-cost preserving sketches* [20]. These show that by sketching the matrix $\mathbf{A}$ on the left and right by two small random sign matrices $\mathbf{S} \in \mathbb{R}^{O(k/\varepsilon^2)\times m}$ and $\mathbf{T} \in \mathbb{R}^{n\times O(k/\varepsilon^2)}$, with $O(sk^2/\varepsilon^4)$ words of communication, the server can learn $\tilde{\mathbf{A}} = \mathbf{SAT}$ which is a "sketch" of $\mathbf{A}$, and critically, that it suffices to compute the best rank-k approximation $\tilde{\mathbf{A}}_k$ to $\tilde{\mathbf{A}}$. Suppose the SVD of $\tilde{\mathbf{A}}_k = \mathbf{U}_{\tilde{\mathbf{A}}_k}\mathbf{\Sigma}_{\tilde{\mathbf{A}}_k}\mathbf{V}_{\tilde{\mathbf{A}}_k}^{\mathrm{T}}$. Then the server can learn $\mathbf{X} = \mathbf{ATV}_{\tilde{\mathbf{A}}_k}$ by a second round of communication which needs $O(skm)$ words. We then prove $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$ where $\mathbf{U}$ is an orthonormal basis of *span*$(\mathbf{X})$. Notice that $rank(\mathbf{U}) \leq k$. Thus $\mathbf{U}$ is exactly what we want. Section 4.3 gives details of the protocol.

A technical obstacle in the analysis above is that it may require a large number of machine words to specify the entries of $\mathbf{V}_{\tilde{\mathbf{A}}_k}$, even if each entry in $\mathbf{A}$ is only a single word. In fact, one can show (we omit the details) that the singular values of $\tilde{\mathbf{A}}_k$ can be exponentially large in k/ε, which means one would need to round the entries of $\mathbf{V}_{\tilde{\mathbf{A}}_k}$ to an additive exponentially small precision, which would translate to an undesirable $sm \cdot \text{poly}(k/\varepsilon)$ bits of communication.

To counter this, we use the *smoothed analysis* of Tao and Vu [44] to argue that if we add small random Bernoulli noise to $\mathbf{A}$, then its minimum singular value becomes inverse polynomial in n. Moreover, if the rank of $\mathbf{A}$ is at least $2k$ and under the assumption that the entries of $\mathbf{A}$ are representable with a single machine word (so we can identify them with integers in magnitude at most $\text{poly}(nms/\varepsilon)$ by scaling), then we can show the additional noise preserves relative error approximation. To our knowledge, this is the first application of this smoothing technique to distributed linear algebra algorithms. Section 5 discusses the details of this approach.

On the other hand, if the rank of $\mathbf{A}$ is smaller than $2k$, the smoothing technique need not preserve relative error. In this case though, we can learn a basis $\mathbf{C} \in \mathbb{R}^{m\times O(k)}$ for the column span of $\mathbf{A}$ by multiplying by a pseudorandom matrix based on Vandermonde matrices. Since $\mathbf{C}$ has at most $2k$ columns, we can efficiently communicate it to all servers using $O(skm)$ communication. At this point we set up the optimization problem $\min_{\text{rank}(\mathbf{X})\leq k} \|\mathbf{C}\mathbf{X}\mathbf{C}^{\mathrm{T}}\mathbf{A} - \mathbf{A}\|_{\mathrm{F}}$. The server cannot learn $\mathbf{C}^{\mathrm{T}}\mathbf{A}$ and $\mathbf{A}$ directly, as this would be $\Omega(nm)$ words, but we can choose additional "sketching matrices" $\mathbf{T}_{left}$ and $\mathbf{T}_{right}$ to instead solve

$$\min_{\text{rank}(\mathbf{X})\leq k} \|\mathbf{T}_{left}(\mathbf{C}\mathbf{X}\mathbf{C}^{\mathrm{T}}\mathbf{A} - \mathbf{A})\mathbf{T}_{right}\|_{\mathrm{F}},$$

which is a generalization of the subspace embedding technique to "affine embeddings" [45]. Note that the server learns $\mathbf{T}_{left}\mathbf{C}\mathbf{X}\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{T}_{right}$ and $\mathbf{T}_{left}\mathbf{A}\mathbf{T}_{right}$, which are small matrices, and can be sent to each machine. Each machine locally solves for the rank-k solution $\mathbf{X}_*$ minimizing the following quantity $\|\mathbf{T}_{left}\mathbf{C}\mathbf{X}\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{T}_{right} - \mathbf{T}_{left}\mathbf{A}\mathbf{T}_{right}\|_{\mathrm{F}}$. Finally, each machine can find the same $m \times k$ orthonormal basis and output it without further communication.

Sparse Matrices in the Column Partition Model.

Our algorithm for sparse matrices is technically simpler than that in the arbitrary partition model. Section 7.2 presents

	Upper bounds		Lower bounds	
One-pass turnstile, Def. 8	$O(mk\varepsilon^{-1} + \mathrm{poly}(k, \varepsilon^{-1}))$	(Theorem 9)	$\Omega(mk\varepsilon^{-1})$ bits	[45]
One-pass turnstile, factorization, Def. 9	$O((n+m)k\varepsilon^{-1} + \mathrm{poly}(k, \varepsilon^{-1}))$	(Theorem 10)	$\Omega((n+m)k\varepsilon^{-1})$ words	[17]
Two-pass turnstile, Def. 8	$O(mk + \mathrm{poly}(k, \varepsilon^{-1}))$	(Theorem 11)	$\Omega(mk)$ bits	[45]

Table 2: **Space upper/lower bounds for the Streaming PCA problem.**

	Upper bounds		Lower bounds	
Definition 5	$O(sk\phi\varepsilon^{-1})$	(Theorem 13)	$\Omega(sk\phi\varepsilon^{-1})$	(Theorem 16)
Definition 6	$O\left(sk\phi\varepsilon^{-1} + s \cdot \mathrm{poly}(k/\varepsilon)\right)$	(Theorem 13)	$\Omega(sk\phi\varepsilon^{-1})$	(Corollary 3)

Table 3: **Communication upper/lower bounds for the CSSP problems of Definitions 5 and 6.**

a distributed PCA algorithm using $O((\phi ks\varepsilon^{-1}) + \mathrm{poly}(sk/\varepsilon))$ words of communication. Our idea in order to take advantage of the sparsity in the input matrix $\mathbf{A}$ is to select and transfer to the coordinator a small set of "good" columns from each sub-matrix $\mathbf{A}_i$. Specifically, we design an algorithm that first computes, in a distributed way, a matrix $\tilde{\mathbf{C}} \in \mathbb{R}^{m \times c}$ with $c = O(k\varepsilon^{-1})$ columns of $\mathbf{A}$, and then finds $\mathbf{U} \in span(\tilde{\mathbf{C}})$ using a distributed, communication-efficient algorithm developed in [30]. The matrix $\tilde{\mathbf{C}}$ is constructed using optimal algorithms for column sampling [13, 22], extended properly to the distributed case.

As mentioned, our algorithm actually solves the distributed column subset selection problem. It builds on results for column subset selection in the batch setting [13, 22], but gives a novel analysis showing that in the distributed setting one can find columns as good as in the batch setting. To the best of our knowledge, only heuristics for distributed column subset selection were known [23]. We also optimize the time complexity of our algorithm, see Table 4.

Turnstile Streaming Model.

Our one-pass streaming PCA algorithm starts by generating two random sign matrices $\mathbf{S} \in \mathbb{R}^{O(k/\varepsilon) \times n}$ and $\mathbf{R} \in \mathbb{R}^{m \times O(k/\varepsilon)}$. We can show, $\min_{\mathrm{rank}(\mathbf{X}) \le k} \|\mathbf{ARXSA} - \mathbf{A}\|_{\mathrm{F}}^2 \le (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$ The intuition of the above is as follows. Suppose the SVD of $\mathbf{A}_k$ is $\mathbf{U}_{\mathbf{A}_k} \mathbf{\Sigma}_{\mathbf{A}_k} \mathbf{V}_{\mathbf{A}_k}^{\mathrm{T}}$ where $\mathbf{A}_k$ is the best rank-k approximation matrix to $\mathbf{A}$. Since $\mathbf{X}$ can be chosen as $\mathbf{\Sigma}_{\mathbf{A}_k} \mathbf{V}_{\mathbf{A}_k}$, we have $\min_{rank(\mathbf{X}) \le k} \|\mathbf{U}_{\mathbf{A}_k}\mathbf{X} - \mathbf{A}\|_{\mathrm{F}}^2 = \|\mathbf{A}_k - \mathbf{A}\|_{\mathrm{F}}^2$. As shown in [17], $\mathbf{S}$ can provide a sketch for the regression problem with $O(k/\epsilon)$ rows. Thus, $\|\mathbf{U}_{\mathbf{A}_k}\tilde{\mathbf{X}} - \mathbf{A}\|_{\mathrm{F}}^2 \le (1+\varepsilon) \cdot \min_{rank(\mathbf{X}) \le k} \|\mathbf{U}_{\mathbf{A}_k}\mathbf{X} - \mathbf{A}\|_{\mathrm{F}}^2$ where $\tilde{\mathbf{X}} = \arg\min_{rank(\mathbf{X}) \le k} \|\mathbf{SU}_{\mathbf{A}_k}\mathbf{X} - \mathbf{SA}\|_{\mathrm{F}}^2$. Notice that $\tilde{\mathbf{X}}$ is in the row space of $\mathbf{SA}$ and has rank k. Then we focus on the regression problem: $\min_{\mathrm{rank}(\mathbf{X}) \le k} \|\mathbf{XSA} - \mathbf{A}\|_{\mathrm{F}}^2$. The main observation is that we can write this as $\min_{\mathrm{rank}(\mathbf{Y}) \le k} \|\mathbf{YWSA} - \mathbf{A}\|_{\mathrm{F}}^2$, where $\mathbf{WSA}$ has the same row space as $\tilde{\mathbf{X}}$ and is a $k \times n$ matrix. By doing this, it follows that if $\mathbf{R}$ has $O(k/\epsilon)$ columns, we can again sketch the problem by multiplying by $\mathbf{R}$ on the right. Without this observation, note that $\mathbf{R}$ would have to have $O(k/\epsilon^2)$ columns to apply the technique in [17], since the matrix in the regression problem in [17] would have rank $O(k/\epsilon)$ instead of k. Analogously, $\mathbf{Y}$ is in the column span of $\mathbf{AR}$. Thus we can optimize

$$\min_{\mathbf{Y}', \mathbf{W}} \|\mathbf{ARY}'\mathbf{WSA} - \mathbf{A}\|_{\mathrm{F}}^2,$$

which we can write as $\min_{\mathrm{rank}(\mathbf{X}) \le k} \|\mathbf{ARXSA} - \mathbf{A}\|_{\mathrm{F}}^2$.

One problem is that it is too expensive to store the entire matrix $\mathbf{A}$. Thus, we sketch on the left and right by $\mathbf{T}_{left}$ and $\mathbf{T}_{right}$. Since each of $\mathbf{T}_{left}\mathbf{AR}$, $\mathbf{SAT}_{right}$ and $\mathbf{T}_{left}\mathbf{AT}_{right}$ can be maintained in small space,

$$\mathbf{X}_* = \arg \min_{\mathrm{rank}(\mathbf{X}) \le k} \|\mathbf{T}_{left}(\mathbf{ARXSA} - \mathbf{A})\mathbf{T}_{right}\|_{\mathrm{F}}^2$$

can be constructed after one pass over the stream. Additionally, if $\mathbf{AR}$ is also maintained in the algorithm, we can get $\mathbf{U} \in \mathbb{R}^{m \times k}$ of which columns are an orthonormal basis of $span(\mathbf{ARX}_*)$ satisfying $\|\mathbf{A} - \mathbf{UU}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \le (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$. Furthermore, if $\mathbf{SA}$ is also maintained, since it suffices to compute the SVD of $\mathbf{X}_* = \mathbf{U}_{\mathbf{X}_*}\mathbf{\Sigma}_{\mathbf{X}_*}\mathbf{V}_{\mathbf{X}_*}^{\mathrm{T}}$, $\mathbf{T} = \mathbf{T}_{left}\mathbf{ARU}_{\mathbf{X}_*}$ and $\mathbf{K} = \mathbf{V}_{\mathbf{X}_*}^{\mathrm{T}}\mathbf{SAT}_{right}$ can be constructed in $O((n+m)k)$ words of space. Therefore, the algorithm can compute $\mathbf{A}_k^* = \mathbf{T\Sigma}_{\mathbf{X}_*}\mathbf{K} = \mathbf{ARX}_*\mathbf{SA}$ in $O((n+m)k)$ words of space. The algorithm then can output $\mathbf{A}_k^*$ with total space $O((n+m)k/\varepsilon + \mathrm{poly}(k/\varepsilon))$ words (see Theorem 10).

Our two-pass streaming PCA algorithm is just an implementation of our distributed PCA algorithm in the streaming model.

1.2.2 Lower Bounds

Table 3 summarizes the matching communication lower bounds that we obtain for distributed column-based matrix reconstruction. Theorem 16 proves a lower bound for the problem in Definition 5; then, a lower bound for the problem of Definition 6 follows immediately since this is a harder problem (see Corollalry 3).

Distributed Column Subset Selection.

Our most involved lower bound argument is in showing the tightness for the distributed column subset selection problem, and is given in Theorem 16. To illustrate the ideas, suppose $k = 1$. We start with the canonical hard matrix for column subset selection, namely, an $m \times m$ matrix $\mathbf{A}$ whose first row is all ones, and remaining rows are a subset of the identity matrix. Intuitively the best rank-1 approximation is very aligned with the first row of $\mathbf{A}$. However, given only $o(1/\varepsilon)$ columns of the matrix, there is not a vector in the span which puts a $(1-\varepsilon)$-fraction of its mass on the first coordinate, so $\Omega(1/\varepsilon)$ columns are needed. Such a matrix has $\phi = 2$.

Obtaining a lower bound for general ϕ involves creating a large net of such instances. Suppose we set $m = \phi$ and consider $\tilde{\mathbf{L}}\mathbf{A}$, where $\tilde{\mathbf{L}}$ is formed by choosing a random $\phi \times \phi$ orthonormal matrix, and then rounding each entry to the nearest integer multiple of $1/\mathrm{poly}(n)$. We can show that $o(1/\varepsilon)$ columns of $\tilde{\mathbf{L}}\mathbf{A}$ do not span a rank-1 approximation to $\tilde{\mathbf{L}}\mathbf{A}$. We also need a lemma which shows that if we take two independently random orthonormal matrices, and round their entries to integer multiples of $1/\mathrm{poly}(n)$, then these matrices are extremely unlikely to share a constant fraction of columns. The idea then is that if one server holds $\tilde{\mathbf{L}}\mathbf{A}$, and every other server holds the zero matrix, then every server needs to output the same subset of columns of $\tilde{\mathbf{L}}\mathbf{A}$. By construction of $\mathbf{A}$, each column of $\tilde{\mathbf{L}}\mathbf{A}$ is the sum of the first column of $\tilde{\mathbf{L}}$ and an arbitrary column of $\tilde{\mathbf{L}}$, and since we can assume all servers know the first column of $\tilde{\mathbf{L}}$ (which involves a negligible $O(s\phi)$ amount of communication), this implies that each server must learn $\Omega(1/\varepsilon)$ columns of $\tilde{\mathbf{L}}$. The probability that random discretized orthonormal matrices share $\Omega(1/\varepsilon)$ columns is sufficiently small that we can choose a large enough net for these columns to identify $\tilde{\mathbf{L}}$ in that net, requiring $\Omega(\phi/\varepsilon)$ bits of communication per server, or $\Omega(s\phi/\varepsilon)$ words in total. The argument for general k can be viewed as a "direct sum theorem", giving $\Omega(sk\phi/\varepsilon)$ words of communication in total.

Dense Matrices in the Column Partition Model.

Our optimal lower bound for dense matrices is technically simpler. It gives an $\Omega(skm)$ word communication lower bound in the column partition model for dense matrices. Theorem 15 argues that there exists an $m \times n$ matrix $\mathbf{A}$ such that for this $\mathbf{A}$, any $k \leq .99m$, and any error parameter C with $1 < C < \mathrm{poly}(skm)$, if there exists a protocol to construct an $m \times k$ matrix $\mathbf{U}$ satisfying $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq C \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$, with constant probability, then this protocol has communication cost at least $\Omega(skm)$ words. This lower bound is proven for a matrix $\mathbf{A}$ that has k fully dense columns. We should note that the lower bound holds even if each machine only outputs the projection of $\mathbf{A}_i$ onto $\mathbf{U}$, i.e., $\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}_i$, rather than $\mathbf{U}$ itself. Note that this problem is potentially easier, since given $\mathbf{U}$, a machine could find $\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}_i$.

The intuition of the proof is that machine 1 has a random matrix $\mathbf{A}_1 \in \mathbb{R}^{m\times k}$ with orthonormal columns (rounded to integer multiples of $1/\mathrm{poly}(n)$), while all other machines have a very small multiple of the identity matrix. When concatenating the columns of these matrices, the best k-dimensional subspace to project the columns onto must be very close to $\mathbf{A}_1$ in spectral norm. Since machine i, $i > 1$, has tiny identity matrices, after projection it obtains a projection matrix very close to the projection onto the column span of $\mathbf{A}_1$. By choosing a net of $m \times k$ orthonormal matrices, all of which pairwise have high distance in spectral norm, each machine can reconstruct a random element in this net, which requires lots of information, and hence communication.

Our lower bound of Theorem 15 also implies a lower bound of $\Omega(sk\phi)$ words for matrices in which the column sparsity is ϕ (see Corollary 2) as a simple corollary.

1.3 Road Map

We discuss prior results on distributed PCA algorithms in Section 2. Section 3 introduces the notation and basic results from linear algebra that we use in our algorithms. Section 4 presents our distributed PCA algorithm for arbitrary matrices for Definition 3. The communication of the algorithm in this section can be stated only in terms of "real numbers". We resolve this issue in Section 5 where a modified algorithm has communication cost bounded in terms of machine words. Section 6 discusses space-optimal PCA methods in the streaming model of computation. Section 7 presents a distributed PCA algorithm for sparse matrices in the column partition model, while Section 8 extends this algorithm to a faster distributed PCA algorithm for sparse matrices. Section 9 presents communication lower bounds.

Proofs are omitted due to space constraints; we refer the reader to [16] for the full version of our paper.

2. RELATED WORK

Distributed PCA (or distributed SVD) algorithms have been investigated for a long time. One line of work, developed primarily within the numerical linear algebra literature, studies such algorithms from the perspective of parallelizing existing standard SVD algorithms without sacrificing accuracy. This approach aims at high accuracy implementations with the least possible communication cost. The distributed models of computation go typically beyond the column-partition model and arbitrary-partition model that we study in this paper [38]. An extensive survey of this line of work is out of the scope of our paper; we refer the reader to [29, 42, 28] and references therein for more details as well as to popular software for distributed SVD such ScaLAPACK [10] and Elemental [38].

Another line of work for distributed PCA algorithms has emerged within the machine learning and datamining communities. Such algorithms have been motivated by the need to apply SVD or PCA to extremely large matrices encoding enormous amounts of data. The algorithms in this line of research are typically heuristic approaches that work well in practice but come with no rigorous theoretical analysis. We refer the reader to [39, 34, 5, 11] for more details.

Finally, distributed PCA algorithms in the column-partition model have been recently studied within the theoretical computer science community. Perhaps the more intuitive algorithm for PCA in this model appeared in [24, 6]: first, a number of left singular vectors and singular values are computed in each machine; then, the server collects those singular vectors and concatenates them column-wise in a new matrix and then it computes the top k left singular vectors of this "aggregate" matrix. It is shown in [24, 6] that if the number of singular vectors and singular values in the first step is $O(k\varepsilon^{-1})$, then, the approximation error in Frobenius norm is at most $(1 + \varepsilon)$ times the optimal Frobenius norm error; the communication cost is $O(skm\varepsilon^{-1})$ *real numbers* because each of the s machines sends $O(k\varepsilon^{-1})$ singular vectors of dimension m; unfortunately, it is unclear how one can obtain a communication cost in terms of words/bits. A different algorithm with the same communication cost (only in terms of real numbers since a word/bit communication bound remained unexplored) is implicit in [33, 26] (see Theorem 3.1 in [26] and the discussion in Section 2.2 in [33]). Kannan, Vempala and Woodruff proposed the arbitrary-partition model in [30]. They developed a $(1 + \varepsilon)$ Frobenius norm error algorithm with communication $O\left(skm\varepsilon^{-1} + sk^2\varepsilon^{-4}\right)$ words. Bhojanapalli, Jain, and Sanghavi in [9] developed an algorithm that provides a bound with respect to the spectral norm. Their algorithm is based on sampling elements from the input matrix, but the communication cost is prohibitive if n is large. The cost also depends on the condition number of $\mathbf{A}$. Moreover, to implement the algorithm, one needs to know $\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}$. Finally, [32] discussed a distributed implementation of the "orthogonal iteration" to compute eigenvectors of graphs. The model they consider is different, but perhaps their protocol could be extended to our model.

3. PRELIMINARIES

DEFINITION 1. *(Arbitrary-partition model) An $m \times n$ matrix $\mathbf{A}$ is arbitrarily partitioned into s matrices $\mathbf{A}_i \in \mathbb{R}^{m\times n}$, i.e., for $i = 1, 2, \ldots, s$: $\mathbf{A} = \sum_{i=1}^{s} \mathbf{A}_i$. There are s machines, and the i-th machine has $\mathbf{A}_i$ as input. There is also another machine, to which we refer to as the "server", which acts as the central coordinator. The model only allows communication between the machines and the server. The communication cost of an algorithm in this model is defined as the total number of words transferred between the machines and the server, where we assume each word is $O(\log(nms/\varepsilon))$ bits.*

DEFINITION 2. *(Column-partition model) An $m \times n$ matrix $\mathbf{A}$ is partitioned arbitrarily column-wise into s blocks $\mathbf{A}_i \in \mathbb{R}^{m\times w_i}$,*

Reference	$\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_2 \leq$	$\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq$	δ	Communication cost	Total number of arithmetic operations
Implicit in [24]	-	$(1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$	0	-	$O\left(mn\min\{m,n\} + msk\varepsilon^{-1}\min\{m, sk\varepsilon^{-1}\}\right)$
Theorem 2 in [6]	-	$(1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$	0	-	$O\left(mn\min\{m,n\} + msk\varepsilon^{-1}\min\{m, sk\varepsilon^{-1}\}\right)$
Theorem 6 in [6]	-	$(1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$	> 0	-	$O\left(\mathrm{nnz}(\mathbf{A}) + s\left(\frac{m^3k}{\varepsilon^4} + \frac{k^2m^2}{\varepsilon^6}\right)\log\left(\frac{m}{\varepsilon}\right)\log\left(\frac{sk}{d\varepsilon}\right)\right)$
Implicit in [33, 26]	-	$(1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$	0	-	$O(mnk\varepsilon^{-2})$
Thm 1.1 in [30]*	-	$(1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$	$O(1)$	$O\left(skm\varepsilon^{-1} + sk^2\varepsilon^{-4}\right)$	$O(poly(m,n,k,s,\varepsilon^{-1}))$
Thm 5.1 in [9]	$\|\mathbf{A} - \mathbf{A}_k\|_2 + \Gamma$	-	> 0	$O\left(sm + nk^5\varepsilon^{-2}\Delta\right)$	$O\left(\mathrm{nnz}(\mathbf{A}) + \delta^{-1}nk^5\varepsilon^{-2}\sigma_1^2(\mathbf{A})\sigma_k^{-2}(\mathbf{A})\right)$
Remark p. 11 [9]	$\|\mathbf{A} - \mathbf{A}_k\|_2 + \Gamma$	-	> 0	$O\left(sm + nk^3\varepsilon^{-2}\Delta\right)$	$O\left(\mathrm{nnz}(\mathbf{A}) + \delta^{-1}nk^5\varepsilon^{-2}\sigma_1^2(\mathbf{A})\sigma_k^{-2}(\mathbf{A})\right)$
Theorem 8*	-	$(1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$	$O(1)$	$O(skm + sk^3\varepsilon^{-5})$	$O\left(mnk\varepsilon^{-2} + mk^2\varepsilon^{-4} + poly(k\varepsilon^{-1})\right)$.
Theorem 12	-	$(1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$	$O(1)$	$O(sk\phi\varepsilon^{-1} + sk^2\varepsilon^{-4})$	$O\left(mn\min\{m,n\} + mns \cdot poly(k, 1/\varepsilon)\right)$
Theorem 13	-	$(1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$	$O(1)$	$O(sk\phi\varepsilon^{-1} + sk^3\varepsilon^{-5})$	$O\left(\mathrm{nnz}(\mathbf{A}) \cdot \log^2(\frac{ns}{\delta}) + (m+n) \cdot s \cdot poly(\frac{k}{\varepsilon}\log(\frac{ns}{\delta})\right)$.

Table 4: **Distributed PCA Algorithms in the column-partition model with s machines (see Definition 2). $*$ indicates that the algorithm can also be applied in the arbitrary partition model (see Definition 1). $\mathbf{A} \in \mathbb{R}^{m \times n}$ has rank ρ, $\mathbf{U} \in \mathbb{R}^{m \times k}$ with $k < \rho$ is orthonormal, $0 < \varepsilon < 1$, and each column of $\mathbf{A}$ contains at most $\phi \leq m$ non-zero elements. δ is the failure probability. Finally, for notational convenience let $\Gamma := \varepsilon\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}$, $\Delta := \sigma_1^2(\mathbf{A})\sigma_k^{-2}(\mathbf{A})\log^2\left(\|\mathbf{A}\|_2\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^{-1}\varepsilon^{-1}\right)$.**

i.e., for $i = 1, 2, \ldots, s$: $\mathbf{A} = \begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2 & \ldots & \mathbf{A}_s\end{pmatrix}$. Here, $\sum w_i = n$. There are s machines, and the i-th machine has $\mathbf{A}_i$ as input. There is also another machine, to which we refer to as the "server", which acts as the central coordinator. The model only allows communication between the machines and the server. The communication cost of an algorithm in this model is defined as the total number of words transferred between the machines and the server, where we assume each word is $O(\log(nms/\varepsilon))$ bits.

DEFINITION 3. *(Distributed PCA - arbitrary partition) Given an $m \times n$ matrix $\mathbf{A}$ arbitrarily partitioned into s matrices $\mathbf{A}_i \in \mathbb{R}^{m\times n}$ $(i: 1, 2, \ldots, s)$: $\mathbf{A} = \sum_{i=1}^{s} \mathbf{A}_i$, a rank parameter $k <$ rank$(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$, design an algorithm in the model of Definition 1 which, upon termination, leaves on each machine a matrix $\mathbf{U} \in \mathbb{R}^{m\times k}$ with k orthonormal columns such that $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$, and the communication cost of the algorithm is as small as possible.*

DEFINITION 4. *(Distributed PCA - column partition) Given an $m \times n$ matrix $\mathbf{A}$ partitioned column-wise into s arbitrary blocks $\mathbf{A}_i \in \mathbb{R}^{m\times w_i}$ $(i: 1, 2, \ldots, s)$: $\mathbf{A} = \begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2 & \ldots & \mathbf{A}_s\end{pmatrix}$, a rank parameter $k <$ rank$(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$, design an algorithm in the model of Definition 2 which, upon termination, leaves on each machine a matrix $\mathbf{U} \in \mathbb{R}^{m\times k}$ with k orthonormal columns such that $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$, and the communication cost of the algorithm is as small as possible.*

DEFINITION 5. *(Distributed Column Subset Selection) Given an $m \times n$ matrix $\mathbf{A}$ partitioned column-wise into s arbitrary blocks $\mathbf{A}_i \in \mathbb{R}^{m\times w_i}$ $(i: 1, 2, \ldots, s)$: $\mathbf{A} = \begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2 & \ldots & \mathbf{A}_s\end{pmatrix}$, a rank parameter $k <$ rank$(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$, design an algorithm in the model of Definition 2 that, upon termination, leaves on each machine a matrix $\mathbf{C} \in \mathbb{R}^{m\times c}$ with $c < n$ columns of $\mathbf{A}$ such that $\|\mathbf{A} - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$, and 1. The number of selected columns c is as small as possible. 2. The communication cost of the algorithm is as small as possible.*

DEFINITION 6. *(Distributed CSS - rank k subspace) Given an $m \times n$ matrix $\mathbf{A}$ partitioned column-wise into s arbitrary blocks $\mathbf{A}_i \in \mathbb{R}^{m\times w_i}$ $(i: 1, 2, \ldots, s)$: $\mathbf{A} = \begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2 & \ldots & \mathbf{A}_s\end{pmatrix}$, a rank parameter $k <$ rank$(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$, design an algorithm in the model of Definition 2 that, upon termination, leaves on each machine a matrix $\mathbf{C} \in \mathbb{R}^{m\times c}$ with $c < n$ columns of $\mathbf{A}$ and a matrix $\mathbf{U} \in \mathbb{R}^{m\times k}$ with k orthonormal columns with $\mathbf{U} \in$ span$(\mathbf{C})$, such that $\|\mathbf{A} - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}\|_{\mathrm{F}}^2 \leq \|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$, and 1. The number of selected columns c is as small as possible. 2. The communication cost of the algorithm is as small as possible.*

DEFINITION 7. *(Streaming model for PCA) Let all the entries in $\mathbf{A} \in \mathbb{R}^{m\times n}$ initially be zero. In the streaming model of computation, there is a stream of update operations such that the q^{th} operation has the form (i_q, j_q, x_q) which indicates that $\mathbf{A}_{i_q, j_q}$ should be incremented by x_q where $i_q \in \{1, ..., m\}, j_q \in \{1, ..., n\}, x_q \in \mathbb{R}$. An algorithm is allowed a single pass over the stream. At the end of the stream the algorithm stores some information regarding $\mathbf{A}$ which we call a "sketch" of $\mathbf{A}$. The space complexity of an algorithm in this model is defined as the total number of words required to describe the information the algorithm stores during the stream including the sketch. Each word is $O(\log(nms/\varepsilon))$ bits.*

DEFINITION 8. *(The Streaming PCA Problem) Given an $m \times n$ matrix $\mathbf{A}$, a rank parameter $k <$ rank$(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$, design a minimal space algorithm that finds a sketch of $\mathbf{A}$ in the streaming model (see Definition 7) and using only this sketch outputs $\mathbf{U} \in \mathbb{R}^{m\times k}$ with k orthonormal columns with $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$.*

DEFINITION 9. *(The Streaming PCA Problem (factorization)) Given an $m \times n$ matrix $\mathbf{A}$, a rank parameter $k <$ rank$(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$, design an algorithm that, using as little space as possible, first finds a sketch of $\mathbf{A}$ in the streaming model (see Definition 7) and then, using only this sketch outputs a factorization of $\mathbf{A}_k^* \in \mathbb{R}^{m\times n}$ with $rank(\mathbf{A}_k^*) \leq k$ such that*

$$\|\mathbf{A} - \mathbf{A}_k^*\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2.$$

Notation.

$\mathbf{A}, \mathbf{B}, \ldots$ are matrices; $\mathbf{a}, \mathbf{b}, \ldots$ are column vectors. $\mathbf{I}_n$ is the $n \times n$ identity matrix; $\mathbf{0}_{m\times n}$ is the $m \times n$ matrix of zeros; $\mathbf{1}_n$ is the $n \times 1$ vector of ones; $\boldsymbol{e}_i$ is the standard basis (whose dimensionality will be clear from the context): the ith element of $\boldsymbol{e}_i$ is one and the rest are zeros. $\mathbf{A}^{(i)}$ and $\mathbf{A}_{(j)}$ denote the ith column and jth row of $\mathbf{A}$, respectively. $\mathbf{A}_{ij}$ is the (i, j)th entry in $\mathbf{A}$.

Sampling Matrices.

Let matrix $\mathbf{A} = [\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(n)}] \in \mathbb{R}^{m\times n}$, and let matrix $\mathbf{C} = [\mathbf{A}^{(i_1)}, \ldots, \mathbf{A}^{(i_c)}] \in \mathbb{R}^{m\times c}$ consist of $c < n$ columns of $\mathbf{A}$. Note that we can write $\mathbf{C} = \mathbf{A}\boldsymbol{\Omega}$, where the *sampling matrix* is $\boldsymbol{\Omega} = [\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_c}] \in \mathbb{R}^{n\times c}$ (here $\mathbf{e}_i$ are the standard basis vectors in $\mathbb{R}^n$). If $\mathbf{D} \in \mathbb{R}^{c\times c}$ is a diagonal matrix, then $\mathbf{A}\boldsymbol{\Omega}\mathbf{D}$ contains c columns of $\mathbf{A}$ rescaled with the corresponding elements in $\mathbf{D}$. We abbreviate $\mathbf{S} := \boldsymbol{\Omega}\mathbf{D}$, hence the matrix $\mathbf{S} \in \mathbb{R}^{n\times c}$ "samples" and "rescales" c columns from $\mathbf{A}$.

Matrix Norms.

The Frobenius and the spectral matrix-norms: $\|\mathbf{A}\|_{\mathrm{F}}^2 = \sum_{i,j}\mathbf{A}_{ij}^2$; $\|\mathbf{A}\|_2 = \sup_{\|\mathbf{x}\|_2=1}\|\mathbf{A}\mathbf{x}\|_2$. $\|\mathbf{A}\|_\xi$ is used if a result holds for both norms $\xi = 2$ and $\xi = \mathrm{F}$. The standard sub-multiplicativity property for matrix norms implies that for any $\mathbf{A}$ and $\mathbf{B}$: $\|\mathbf{AB}\|_\xi \leq \|\mathbf{A}\|_\xi \cdot \|\mathbf{B}\|_\xi$. The triangle inequality for matrix norms implies that $\|\mathbf{A} + \mathbf{B}\|_\xi \leq \|\mathbf{A}\|_\xi + \|\mathbf{B}\|_\xi$. A version of the triangle inequality for the norms squared is: $\|\mathbf{A} + \mathbf{B}\|_\xi^2 \leq 2 \cdot \|\mathbf{A}\|_\xi^2 + 2 \cdot \|\mathbf{B}\|_\xi^2$. A version of the matrix Pythagorean theorem is: if $\mathbf{A}^{\mathrm{T}}\mathbf{B}$ is the all-zeros matrix, then $\|\mathbf{A} + \mathbf{B}\|_{\mathrm{F}}^2 = \|\mathbf{A}\|_{\mathrm{F}}^2 + \|\mathbf{B}\|_{\mathrm{F}}^2$. If $\mathbf{V}$ has orthonormal columns, then $\|\mathbf{A}\mathbf{V}^{\mathrm{T}}\|_\xi = \|\mathbf{A}\|_\xi$, for any $\mathbf{A}$. If $\mathbf{P}$ is a symmetric projection matrix (i.e., $\mathbf{P} = \mathbf{P}^{\mathrm{T}}$ and $\mathbf{P}^2 = \mathbf{P}$) then, $\|\mathbf{PA}\|_\xi \leq \|\mathbf{A}\|_\xi$, for any $\mathbf{A}$.

Singular Value Decomposition (SVD) and Moore-Penrose Pseudo-inverse.

The SVD of $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank$(\mathbf{A}) = \rho$ is

$$\mathbf{A} = \underbrace{\begin{pmatrix} \mathbf{U}_k & \mathbf{U}_{\rho-k} \end{pmatrix}}_{\mathbf{U}_\mathbf{A} \in \mathbb{R}^{m \times \rho}} \underbrace{\begin{pmatrix} \mathbf{\Sigma}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_{\rho-k} \end{pmatrix}}_{\mathbf{\Sigma}_\mathbf{A} \in \mathbb{R}^{\rho \times \rho}} \underbrace{\begin{pmatrix} \mathbf{V}_k^{\mathrm{T}} \\ \mathbf{V}_{\rho-k}^{\mathrm{T}} \end{pmatrix}}_{\mathbf{V}_\mathbf{A}^{\mathrm{T}} \in \mathbb{R}^{\rho \times n}},$$

with singular values $\sigma_1 \geq \ldots \sigma_k \geq \sigma_{k+1} \geq \ldots \geq \sigma_\rho > 0$. Here, $k < \rho$. We will use $\sigma_i(\mathbf{A})$ to denote the i-th singular value of $\mathbf{A}$ when the matrix is not clear from the context. The matrices $\mathbf{U}_k \in \mathbb{R}^{m \times k}$ and $\mathbf{U}_{\rho-k} \in \mathbb{R}^{m \times (\rho-k)}$ contain the left singular vectors of $\mathbf{A}$, and, similarly, the matrices $\mathbf{V}_k \in \mathbb{R}^{n \times k}$ and $\mathbf{V}_{\rho-k} \in \mathbb{R}^{n \times (\rho-k)}$ contain the right singular vectors of $\mathbf{A}$. It is well-known that $\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}} = \mathbf{A}\mathbf{V}_k\mathbf{V}_k^{\mathrm{T}} = \mathbf{U}_k\mathbf{U}_k^{\mathrm{T}}\mathbf{A}$ minimizes $\|\mathbf{A} - \mathbf{X}\|_\xi$ over all matrices $\mathbf{X} \in \mathbb{R}^{m \times n}$ of rank at most k. Specifically, $\|\mathbf{A} - \mathbf{A}_k\|_2^2 = \sigma_{k+1}^2(\mathbf{A})$ and $\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2 = \sum_{i=k+1}^{\rho} \sigma_i^2(\mathbf{A})$ (see [27]). $\mathbf{A}^\dagger = \mathbf{V}_\mathbf{A}\mathbf{\Sigma}_\mathbf{A}^{-1}\mathbf{U}_\mathbf{A}^{\mathrm{T}} \in \mathbb{R}^{n \times m}$ denotes the so-called Moore-Penrose pseudo-inverse of $\mathbf{A} \in \mathbb{R}^{m \times n}$ (here $\mathbf{\Sigma}_\mathbf{A}^{-1}$ is the inverse of $\mathbf{\Sigma}_\mathbf{A}$). By the SVD of $\mathbf{A}$ and $\mathbf{A}^\dagger$, $\forall i = 1, \ldots, \rho = $ rank$(\mathbf{A}) = $ rank$(\mathbf{A}^\dagger)$: $\sigma_i(\mathbf{A}^\dagger) = 1/\sigma_{\rho-i+1}(\mathbf{A})$.

The Best Rank k Matrix within a Subspace.

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, let $k < n$ be an integer, and let $\mathbf{V} \in \mathbb{R}^{m \times c}$ with $k < c < n$. $\Pi_{\mathbf{V},k}^{\mathrm{F}}(\mathbf{A}) \in \mathbb{R}^{m \times n}$ is the best rank k approximation to $\mathbf{A}$ in the column span of $\mathbf{V}$. Equivalently, we can write $\Pi_{\mathbf{V},k}^{\mathrm{F}}(\mathbf{A}) = \mathbf{V}\mathbf{X}_{opt}$, where $\mathbf{X}_{opt} = \underset{\mathrm{rank}(\mathbf{X}) \leq k}{\mathrm{argmin}} \|\mathbf{A} - \mathbf{VX}\|_{\mathrm{F}}^2$. Similarly, we use $\Pi_{\mathbf{R},k}^{\mathrm{F}}(\mathbf{A})$ to denote the best rank k approximation within the row space of a given subspace $\mathbf{R}$.

4. DISTRIBUTED PCA IN THE ARBITRARY PARTITION MODEL

This section describes a fast distributed PCA algorithm with total communication $O(msk) + \mathrm{poly}(sk/\epsilon)$ words, which is optimal up to the typically lower-order $\mathrm{poly}(sk/\epsilon)$ factor.

We first present the batch version of the algorithm which offers a new low-rank matrix approximation technique; a specific implementation of this algorithm offers a communication-optimal distributed PCA algorithm. Before presenting all these new algorithms in detail, we present the relevant results from the previous literature.

4.1 Projection-cost Preserving Sketching Matrices

In this section, we recap a notion of sketching matrices which are called "projection-cost preserving sketching matrices". A sketching matrix from this family is a linear matrix transformation and it has the property that for all projections it preserves, up to some error, the difference between the matrix in hand and its projection in Frobenius norm.

Definition 10. *(Projection-cost preserving sketching) We say that $\mathbf{W} \in \mathbb{R}^{n \times \xi}$ is an (ε, k)-projection-cost preserving sketching matrix of $\mathbf{A} \in \mathbb{R}^{m \times n}$, if for all rank-$k$ orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{m \times m}$, it satisfies $(1 - \varepsilon)\|\mathbf{A} - \mathbf{PA}\|_{\mathrm{F}}^2 \leq \|\mathbf{AW} - \mathbf{PAW}\|_{\mathrm{F}}^2 + c \leq (1 + \varepsilon)\|\mathbf{A} - \mathbf{PA}\|_{\mathrm{F}}^2$, where c is a non-negative scalar which only depends on $\mathbf{A}$ and $\mathbf{W}$. We also call $\mathbf{AW}$ an (ε, k)-projection-cost preserving sketch of $\mathbf{A}$.*

Lemma 1. *(PCA via Projection-Cost Sketching [20]) Suppose $\mathbf{W} \in \mathbb{R}^{n \times \xi}$ is an (ε, k)-projection-cost preserving sketching matrix of $\mathbf{A} \in \mathbb{R}^{m \times n}$. Let $\hat{\mathbf{P}}^* = \arg\min_{rank(\mathbf{P}) \leq k} \|\mathbf{AW} - \mathbf{PAW}\|_{\mathrm{F}}^2$. For all $\hat{\mathbf{P}}, \varepsilon'$ satisfying $rank(\hat{\mathbf{P}}) \leq k, \varepsilon' \geq 0$, if*

$$\|\mathbf{AW} - \hat{\mathbf{P}}\mathbf{AW}\|_{\mathrm{F}}^2 \leq (1 + \varepsilon')\|\mathbf{AW} - \hat{\mathbf{P}}^*\mathbf{AW}\|_{\mathrm{F}}^2,$$

then $\|\mathbf{A} - \hat{\mathbf{P}}\mathbf{A}\|_{\mathrm{F}}^2 \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot (1 + \varepsilon')\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$.

[20] also provides several ways to construct projection-cost preserving sketching matrices. Because we mainly consider the communication, we just choose one which can reduce the dimension as much as possible. Furthermore, it is also an oblivious projection-cost preserving sketching matrix.

Lemma 2. *(Dense Johnson-Lindenstrauss matrix) [part of Theorem 12 in [20]] For $\varepsilon < 1$, suppose each entry of $\mathbf{W} \in \mathbb{R}^{n \times \xi}$ is chosen $O(k)$-wise independently and uniformly in $\{1/\sqrt{\xi}, -1/\sqrt{\xi}\}$ where $\xi = O(k\varepsilon^{-2})$ [17]. For any $\mathbf{A} \in \mathbb{R}^{m \times n}$, with probability at least .99, $\mathbf{W}$ is an (ε, k)-projection-cost preserving sketching matrix of $\mathbf{A}$.*

Note, as pointed out to us, it is possible that the $O(k)$-wise independence can be improved, though the communication cost of our algorithm will still be $O(msk) + s \cdot \mathrm{poly}(k/\epsilon)$ after doing this, so we omit this optimization.

4.2 A Batch Algorithm for The Fast Low Rank Approximation of Matrices

In this section, we describe a new method for quickly computing a low-rank approximation to a given matrix. This method does not offer any specific advantages over previous such techniques [40, 19, 14]; however, this new algorithm can be implemented efficiently in the distributed setting (see Section 4.3) and in the streaming model of computation (see Section 6); in fact we are able to obtain communication-optimal and space-optimal results, respectively. For completeness as well as ease of presentation, we first present and analyze the simple batch version of the algorithm. The algorithm uses the dense Johnson-Lindenstrauss matrix of Lemma 2 in order to reduce both dimensions of $\mathbf{A}$, before computing some sort of SVD to a $\mathrm{poly}(k/\varepsilon) \times \mathrm{poly}(k/\varepsilon)$ matrix (see Step 2 in the algorithm below).

Consider the usual inputs: a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, a rank parameter $k <$ rank$(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$. The algorithm below returns an orthonormal matrix $\mathbf{U} \in \mathbb{R}^{m \times k}$ such that $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1 + \varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$.

Algorithm

1. Construct two Johnson-Lindenstrauss matrices $\mathbf{S} \in \mathbb{R}^{\xi_1 \times m}$ and $\mathbf{T} \in \mathbb{R}^{n \times \xi_2}$ with $\xi_1 = O(k\varepsilon^{-2}), \xi_2 = O(k\varepsilon^{-2})$ (see Lemma 2).
2. Construct $\tilde{\mathbf{A}} = \mathbf{SAT}$.

3. Compute the SVD of $\tilde{\mathbf{A}}_k = \mathbf{U}_{\tilde{\mathbf{A}}_k}\mathbf{\Sigma}_{\tilde{\mathbf{A}}_k}\mathbf{V}^{\mathrm{T}}_{\tilde{\mathbf{A}}_k}$ ($\mathbf{U}_{\tilde{\mathbf{A}}_k} \in \mathbb{R}^{\xi_1 \times k}$, $\mathbf{\Sigma}_{\tilde{\mathbf{A}}_k} \in \mathbb{R}^{k \times k}$, $\mathbf{V}_{\tilde{\mathbf{A}}_k} \in \mathbb{R}^{\xi_2 \times k}$).
4. Construct $\mathbf{X} = \mathbf{ATV}_{\tilde{\mathbf{A}}_k}$
5. Compute an orthonormal basis $\mathbf{U} \in \mathbb{R}^{m \times k}$ for $span(\mathbf{X})$ (notice that $rank(\mathbf{X}) \leq k$).

4.3 The Distributed PCA Algorithm

Recall that the input matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is partitioned arbitrarily as: $\mathbf{A} = \sum_i^s \mathbf{A}_i$ for $i = 1 : s$, $\mathbf{A}_i \in \mathbb{R}^{m \times n}$. The idea is to implement the algorithm in Section 4.2 in the distributed setting. Firstly, all the machines can agree with the same $\mathbf{S}$ and $\mathbf{T}$ in the step 1 of the batch algorithm. Then machine i computes $\mathbf{SA}_i\mathbf{T}$ locally, and sends it to the server. Thus the server can learn $\tilde{\mathbf{A}} = \mathbf{SAT}$ and implement the step 3. Next, the server sends the $\mathbf{V}_{\tilde{\mathbf{A}}_k}$ to all machines. Each machine then computes $\mathbf{A}_i\mathbf{TV}_{\tilde{\mathbf{A}}_k}$ and returns the result to the server. Finally, the server can finish the algorithm and send $\mathbf{U}$ to all machines. Notice that $\mathbf{S}$ and $\mathbf{T}$ can be described using a random seed that is $O(k)$-wise independent due to Lemma 2.

THEOREM 6. *(main) The matrix $\mathbf{U} \in \mathbb{R}^{m \times k}$ with k orthonormal columns satisfies w.p. .98:*

$$\|\mathbf{A} - \mathbf{UU}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1 + \varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2. \tag{1}$$

The communication cost of the algorithm is $O(msk + s \cdot \mathrm{poly}(k/\varepsilon))$ "real numbers"; and the running time of the algorithm is of the order $O\left(nmk\varepsilon^{-2} + mk^2\varepsilon^{-4} + \mathrm{poly}(k\varepsilon^{-1})\right)$.

Notice that the communication cost is only given in terms of "real numbers". This is because the entries of $\mathbf{V}_{\tilde{\mathbf{A}}_k}$ could be unbounded (see the discussion regarding the upper bounds in Section 1.2). We resolve this issue in the next section.

5. OBTAINING BIT COMPLEXITY FOR THE DISTRIBUTED PCA ALGORITHM

In this section, we describe how to obtain a communication upper bound in terms of words for the above protocol, where each word is $O(\log(mnsk/\varepsilon))$ bits.

The basic idea is that we have a case analysis depending on the rank of the matrix $\mathbf{A}$. If the rank of $\mathbf{A}$ is less than or equal to $2k$, we follow one distributed protocol and if the rank is at least $2k$ we follow a different protocol. In Section 5.1 and Section 5.2, we describe the algorithm that tests the rank of a distributed matrix, and the PCA protocol for rank of $\mathbf{A}$ less than or equal to $2k$, respectively. Then, in Section 5.3 and Section 5.4 we give the details of the overall algorithm and in Section 5.5 we give its analysis.

5.1 Testing the Rank of a Distributed Matrix

LEMMA 3. *Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a rank parameter $k < rank(\mathbf{A})$, there exists a distributed protocol in the arbitrary partition model to test if the rank of $\mathbf{A}$ is less than or equal to $2k$ using $O(sk^2)$ words of communication and succeeding with probability $1 - \delta$ for an arbitrarily small constant $\delta > 0$.*

This is an immediate implementation of a streaming algorithm due to [17] for testing if an $n \times n$ matrix $\mathbf{A}$ has rank at least $2k$. In that algorithm, there is a fixed $O(nk) \times n$ matrix $\mathbf{H}$ whose entries are integers of magnitude at most $\mathrm{poly}(n)$. The algorithm chooses $4k$ random rows from $\mathbf{H}$. Letting $\mathbf{H}'$ be the $2k \times n$ matrix of the first $2k$ chosen rows, and $\mathbf{H}''$ be the $n \times 2k$ matrix whose columns are the next $2k$ chosen rows, the algorithm just declares that $\mathbf{A}$ has rank at least k iff the rank of $\mathbf{H}'\mathbf{AH}''$ is $2k$. In the distributed model, since machine i can compute $\mathbf{H}'\mathbf{A}_i\mathbf{H}''$ locally, the coordinator can learn $\mathbf{H}'\mathbf{AH}''$ with $O(sk^2)$ words of communication.

5.2 Distributed PCA when $\mathrm{rank}(\mathbf{A}) \leq 2k$

5.2.1 Subsampled Randomized Hadamard Transform and Affine Embeddings

Our algorithms use a tool, known as "Subsampled Randomized Hadamard Transform" or SRHT for short, to implement efficiently fast dimension reduction in large matrices.

The next lemma argues that an SRHT matrix is a so-called "affine embedding matrix". The SRHT is one of the possible choices of Lemma 32 in [19] that will satisfy the lemma, and we just choose it for convenience.

LEMMA 4. *(Affine embeddings - Theorem 39 in [19]) Suppose $\mathbf{G}$ and $\mathbf{H}$ are matrices with m rows, and $\mathbf{G}$ has rank at most r. Suppose $\mathbf{T}$ is a $\xi \times m$ SRHT matrix with $\xi = \tilde{O}(r/\varepsilon^2)$. Then, with probability .99, for all $\mathbf{X}$ simultaneously:*

$$(1-\varepsilon) \cdot \|\mathbf{GX} - \mathbf{H}\|_{\mathrm{F}}^2 \leq \|\mathbf{T}(\mathbf{GX} - \mathbf{H})\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\mathbf{GX} - \mathbf{H}\|_{\mathrm{F}}^2.$$

5.2.2 Generalized Rank-constrained Matrix Approximations

Let $\mathbf{M} \in \mathbb{R}^{m \times n}$, $\mathbf{N} \in \mathbb{R}^{m \times c}$, $\mathbf{L} \in \mathbb{R}^{r \times n}$, and $k \leq c, r$ be integers. Consider the following optimization problem,

$$\mathbf{X}_{opt} \in \underset{\mathbf{X} \in \mathbb{R}^{c \times r},\ \mathrm{rank}(\mathbf{X}) \leq k}{\mathrm{argmin}} \|\mathbf{M} - \mathbf{NXL}\|_{\mathrm{F}}^2.$$

Then, the solution $\mathbf{X}_{opt} \in \mathbb{R}^{c \times r}$ with $\mathrm{rank}(\mathbf{X}_{opt}) \leq k$ that has the minimum $\|\mathbf{X}_{opt}\|_{\mathrm{F}}$ out of all possible feasible solutions is given via $\mathbf{X}_{opt} = \mathbf{N}^{\dagger}\left(\mathbf{U_N U_N^{\mathrm{T}} M V_L V_L^{\mathrm{T}}}\right)_k \mathbf{L}^{\dagger}$. The matrix $\left(\mathbf{U_N U_N^{\mathrm{T}} M V_L V_L^{\mathrm{T}}}\right)_k \in \mathbb{R}^{m \times n}$ of rank at most k denotes the best rank k matrix to $\mathbf{U_N U_N^{\mathrm{T}} M V_L V_L^{\mathrm{T}}} \in \mathbb{R}^{m \times n}$. This result was proven in [25] (see also [41] for the spectral norm version of the problem).

5.2.3 The PCA Protocol

LEMMA 5. *Suppose the rank ρ of $\mathbf{A} \in \mathbb{R}^{m \times n}$ satisfies $\rho \leq 2k$, for some rank parameter k. Then, there is a protocol for the Distributed Principal Component Analysis Problem in the arbitrary partition model using $O(smk + sk^2/\varepsilon^2)$ words of communication and succeeding with probability $1 - \delta$ for an arbitrarily small constant $\delta > 0$.*

The $n \times 2k$ matrix $\mathbf{H}''$ chosen in the protocol of Lemma 3 satisfies that with probability $1 - \delta$, for an arbitrarily small constant $\delta > 0$, the rank of $\mathbf{AH}''$ is equal to the rank of $\mathbf{A}$ if $\rho < 2k$. Hence, as in the protocol of Lemma 3, the coordinator learns the column space of $\mathbf{A}$, which can be described with $O(km)$ words. The coordinator thus communicates this to all machines, using $O(skm)$ total words of communication. Let $\mathbf{C} = \mathbf{AH}''$, which is $m \times 2k$. Since the rank of $\mathbf{C}$ and $\mathbf{A}$ are small, we can sketch on the left and right using affine embeddings $\mathbf{T}_{left}$ and $\mathbf{T}_{right}$. Each machine can then solve the optimization problem

$$\min_{\mathrm{rank}(\mathbf{X}) \leq k} \|\mathbf{T}_{left}\mathbf{CXC}^{\mathrm{T}}\mathbf{AT}_{right} - \mathbf{T}_{left}\mathbf{AT}_{right}\|_{\mathrm{F}}, \tag{2}$$

each obtaining the same $\mathbf{X}_*$ which is the optimal solution to Eqn (2). Due to Lemma 4, $\mathbf{X}_*$ is a $(1+O(\varepsilon))$-approximation to the best solution to $\min_{\text{rank}(\mathbf{X}) \leq k} \|\mathbf{CXC}^{\mathrm{T}}\mathbf{A} - \mathbf{A}\|_{\mathrm{F}}$. Finally, every machine outputs the same orthonormal basis $\mathbf{U} \in \mathbb{R}^{m \times k}$ for $\mathbf{CX}_*$ which satisfies $\|\mathbf{A} - \mathbf{UU}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$.

5.3 Perturbation Technique when $\text{rank}(\mathbf{A}) > 2k$

The idea here is more involved than in the previous subsection and in order to describe the algorithm we need several intermediate results.

5.3.1 *Lower Bounds on Singular Values of Integer-perturbed Matrices*

We describe a perturbation technique for matrices and provide lower bounds for the smallest singular value of the perturbed matrix. We start with a theorem of Tao and Vu.

THEOREM 7. *(Theorem 2.5 of [44]) Let $\mathbf{M}$ be an $n \times n$ matrix with integer entries bounded in magnitude by n^C for a constant $C > 0$. Let $\mathbf{N}_n$ be a matrix with independent entries each chosen to be 1 with probability 1/2, and -1 with probability 1/2. Then, there is a constant $B > 0$ depending on C, for which $\Pr[\|(\mathbf{M} + \mathbf{N}_n)^{-1}\|_2 \geq n^B] \leq 1/n$.*

COROLLARY 1. *Let $\mathbf{M}$ be an $n \times n$ matrix with integer entries bounded in magnitude by n^C for a constant $C > 0$. Let $\mathbf{N}_n$ be a matrix with independent entries each chosen to be $1/n^D$ with probability 1/2, and $-1/n^D$ with probability 1/2, where $D > 0$ is a constant. Then, there is a constant $B > 0$ depending on C and D for which $\Pr[\|(\mathbf{M} + \mathbf{N}_n)^{-1}\|_2 \geq n^B] \leq 1/n$.*

LEMMA 6. *Let $\mathbf{M}$ be an $m \times n$ matrix with integer entries bounded in magnitude by n^C for a constant $C > 0$, and suppose $m \leq n$. Let $\mathbf{N}_{m,n}$ be a matrix with independent entries each chosen to be $1/n^D$ with probability 1/2 and $-1/n^D$ with probability 1/2, where $D > 0$ is a constant. Then, there is a constant $B > 0$ depending on C and D for which $\Pr[\sigma_m(\mathbf{M} + \mathbf{N}_{m,n}) < 1/n^B] \leq 1/n$, where $\sigma_m(\mathbf{M} + \mathbf{N}_{m,n})$ denotes the smallest singular value of $\mathbf{M} + \mathbf{N}_{m,n}$.*

5.4 Description of the Whole Algorithm

We are now ready to describe a PCA protocol whose communication cost can be bounded in terms of machine words (not "real numbers"). As in the algorithm in Section 4.3, we let $\mathbf{B}_i \in \mathbb{R}^{m \times n}$ denote the matrix that arises from $\mathbf{A}_1$ after applying the Bernoulli perturbation technique discussed above in Lemma 6, while $\forall i > 1$, $\mathbf{B}_i$ is equal to $\mathbf{A}_i$. Using this notation, we have $\mathbf{B} := \sum_i \mathbf{B}_i \in \mathbb{R}^{m \times n}$. Notice that $\mathbf{B}$ exactly arises from $\mathbf{A}$ after applying such a perturbation technique.

Input: 1. $\mathbf{A} \in \mathbb{R}^{m \times n}$ arbitrarily partitioned $\mathbf{A} = \sum_i^s \mathbf{A}_i$ for $i = 1 : s$, $\mathbf{A}_i \in \mathbb{R}^{m \times n}$. 2. rank parameter $k < \text{rank}(\mathbf{A})$. 3. accuracy parameter $\varepsilon > 0$.

Algorithm

1. Use the protocol of Lemma 3 with $\delta = 0.01$ to test if the rank of $\mathbf{A}$ is less than $2k$.
2. If rank($\mathbf{A}$) $\leq 2k$, use the protocol of Lemma 5 to find some orthonormal $\mathbf{U} \in \mathbb{R}^{m \times k}$.
3. If rank($\mathbf{A}$) $> 2k$,
   (a) machine 1 locally and independently adds $1/n^D$ with probability 1/2, and $-1/n^D$ with probability 1/2, to each of the entries of $\mathbf{A}_1$, where D is the constant of Lemma 6. Note that this effectively adds the matrix $\mathbf{N}_{m,n}$ of Lemma 6 to the entire matrix $\mathbf{A}$. For notational convenience let $\mathbf{B} = \mathbf{A} + \mathbf{N}_{m,n}$, $\mathbf{B}_1 \in \mathbb{R}^{m \times n}$ be the local perturbed matrix of machine 1 and $\forall i > 1$, $\mathbf{B}_i$ is just equal to $\mathbf{A}_i$.
   (b) Machines agree upon two dense Johnson-Lindenstrauss matrices $\mathbf{S} \in \mathbb{R}^{\xi_1 \times m}$, $\mathbf{T} \in \mathbb{R}^{n \times \xi_2}$ with $\xi_1 = O(k\varepsilon^{-2})$, $\xi_2 = O(k\varepsilon^{-2})$ (see Lemma 2).
   (c) Each machine locally computes $\tilde{\mathbf{B}}_i = \mathbf{SB}_i\mathbf{T}$ and sends $\tilde{\mathbf{B}}_i$ to the server. Server constructs $\tilde{\mathbf{B}} = \sum_i \tilde{\mathbf{B}}_i$.
   (d) Server computes the SVD of $\tilde{\mathbf{B}}_k = \mathbf{U}_{\tilde{\mathbf{B}}_k}\mathbf{\Sigma}_{\tilde{\mathbf{B}}_k}\mathbf{V}_{\tilde{\mathbf{B}}_k}^{\mathrm{T}}$ ($\mathbf{U}_{\tilde{\mathbf{B}}_k} \in \mathbb{R}^{\xi_1 \times k}$, $\mathbf{\Sigma}_{\tilde{\mathbf{B}}_k} \in \mathbb{R}^{k \times k}$, $\mathbf{V}_{\tilde{\mathbf{B}}_k} \in \mathbb{R}^{\xi_2 \times k}$).
   (e) Server rounds each of the entries in $\mathbf{V}_{\tilde{\mathbf{B}}_k}$ to the nearest integer multiple of $1/n^\gamma$ for a sufficiently large constant $\gamma > 0$. Let the matrix after the rounding be $\hat{\mathbf{V}}_{\tilde{\mathbf{B}}_k}$
   (f) Server sends $\hat{\mathbf{V}}_{\tilde{\mathbf{B}}_k}$ to all machines.
   (g) Each machine constructs $\hat{\mathbf{X}}_i = \mathbf{B}_i\mathbf{T}\hat{\mathbf{V}}_{\tilde{\mathbf{B}}_k}$ and sends $\hat{\mathbf{X}}_i$ to the server. Server constructs $\hat{\mathbf{X}} = \sum_i \hat{\mathbf{X}}_i$.
   (h) Server computes an orthonormal basis $\mathbf{U} \in \mathbb{R}^{m \times k}$ for $span(\hat{\mathbf{X}})$ (notice that $rank(\hat{\mathbf{X}}) \leq k$), e.g., with a QR factorization.
   (i) Server sends $\mathbf{U}$ to each machine.

5.5 Main Result

The theorem below analyzes the previous algorithm.

THEOREM 8. *The matrix $\mathbf{U} \in \mathbb{R}^{m \times k}$ with k orthonormal columns satisfies with arbitrarily large constant probability:*

$$\|\mathbf{A} - \mathbf{UU}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq (1 + \varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2. \tag{3}$$

The communication cost is $O(msk + s \cdot \text{poly}(k/\varepsilon))$ words and the running time is $O\left(nmk\varepsilon^{-2} + mk^2\varepsilon^{-4} + \text{poly}(k\varepsilon^{-1})\right)$.

6. STREAMING PRINCIPAL COMPONENT ANALYSIS

In this section, we are interested in computing a PCA of a matrix in the turnstile streaming model. Specifically, there is a *stream* of update operations such that the q^{th} operation has the form (i_q, j_q, x_q) which indicates that the $(i_q, j_q)^{th}$ entry of $\mathbf{A} \in \mathbb{R}^{m \times n}$ should be incremented by x_q where $i_q \in \{1, ..., m\}$, $j_q \in \{1, ..., n\}$, $x_q \in \mathbb{R}$. Initially, $\mathbf{A}$ is zero. In the streaming model of computation, we are allowed only one pass over the update operations, i.e., the algorithm "sees" each update operation one by one and only once. Upon termination, the algorithm should return a matrix $\mathbf{U}$ with k orthonormal columns which is a "good" basis for $span(\mathbf{A})$. In Section 6.1.1 below, we describe an algorithm which gives a space-optimal streaming algorithm. Furthermore, we provide a variation of this algorithm in Section 6.1.2 which can output a factorization of a matrix $\mathbf{A}_k^*$ satisfying $\|\mathbf{A} - \mathbf{A}_k^*\|_{\mathrm{F}}^2 \leq (1 + \varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}}^2$. It meets the space lower bound shown in [17]. In Section 6.2, we relax the problem and we describe a two-pass streaming algorithm which is a version of the algorithm of Section 5.4. Notice that the space complexity of our algorithm is bounded in terms of bits.

Inputs to the algorithms in Section 6.1 and Section 6.2 are a stream of updates $(i_1, j_1, x_1), \ldots, (i_l, j_l, x_l)$, a rank parameter $k < \text{rank}(\mathbf{A})$, and an accuracy parameter $0 < \varepsilon < 1$.

6.1 One-pass Streaming PCA

6.1.1 *The Algorithm which Outputs* $\mathbf{U}$

In the following algorithm, the output should be $\mathbf{U}$ with orthogonal unit columns and satisfying $\|\mathbf{A} - \mathbf{UU}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \leq$

$(1+\varepsilon)\cdot\|\mathbf{A}-\mathbf{A}_k\|_{\mathrm{F}}^2$. Our algorithm uses the following property of random sign matrices.

LEMMA 7. *(Sketch for regression - Theorem 3.1 in [17]) Suppose both of* $\mathbf{A}$ *and* $\mathbf{B}$ *have* m *rows and* $rank(\mathbf{A}) \le k$*. If each entry of* $\mathbf{S} \in \mathbb{R}^{\xi\times m}$ *is* $O(k)$*-wise independently chosen from* $\{-1/\xi, +1/\xi\}$ *where* $\xi = O(k/\varepsilon)$ *and* $\tilde{\mathbf{X}} = \arg\min_{\mathbf{X}} \|\mathbf{SAX}-\mathbf{SB}\|_{\mathrm{F}}^2$*, w.p. at least* 0.99*,* $\|\mathbf{A}\tilde{\mathbf{X}}-\mathbf{B}\|_{\mathrm{F}}^2 \le (1+\varepsilon)\min_{\mathbf{X}}\|\mathbf{AX}-\mathbf{B}\|_{\mathrm{F}}^2$

Algorithm

```
1. Construct random sign sketching matrices $\mathbf{S} \in \mathbb{R}^{\xi_1\times m}$ with $\xi_1 = O(k\varepsilon^{-1})$ and $\mathbf{R} \in \mathbb{R}^{n\times\xi_2}$ with $\xi_2 = O(k\varepsilon^{-1})$ (see Lemma 7)
2. Construct affine embedding matrices $\mathbf{T}_{left} \in \mathbb{R}^{\xi_3\times m}$ and $\mathbf{T}_{right} \in \mathbb{R}^{n\times\xi_4}$ with $\xi_3 = O(k\varepsilon^{-3})$, $\xi_4 = O(k/\varepsilon^{-3})$ (see Section 5.2.1).
3. Initialize all-zeros matrices: $\mathbf{M} \in \mathbb{R}^{\xi_3\times\xi_4}$, $\mathbf{L} \in \mathbb{R}^{\xi_1\times\xi_4}$, $\mathbf{N} \in \mathbb{R}^{\xi_3\times\xi_2}$, $\mathbf{D} \in \mathbb{R}^{m\times\xi_2}$. We will maintain $\mathbf{M},\mathbf{L},\mathbf{N},\mathbf{D}$ such that $\mathbf{M} = \mathbf{T}_{left}\mathbf{A}\mathbf{T}_{right}$, $\mathbf{L} = \mathbf{SAT}_{right}$, $\mathbf{N} = \mathbf{T}_{left}\mathbf{AR}$ and $\mathbf{D} = \mathbf{AR}$.
4. For $(i_q, j_q, x_q) := (i_1, j_1, x_1), ..., (i_l, j_l, x_l)$ (one pass over the stream of update operations)
    (a) For all $i = 1, ..., \xi_3,\ j = 1, ..., \xi_4$, let $\mathbf{M}_{i,j} = \mathbf{M}_{i,j} + (\mathbf{T}_{left})_{i,i_q}\cdot x_q\cdot(\mathbf{T}_{right})_{j_q,j}$.
    (b) For all $i = 1, ..., \xi_1,\ j = 1, ..., \xi_4$, let $\mathbf{L}_{i,j} = \mathbf{L}_{i,j} + \mathbf{S}_{i,i_q}\cdot x_q\cdot(\mathbf{T}_{right})_{j_q,j}$.
    (c) For all $i = 1, ..., \xi_3,\ j = 1, ..., \xi_2$, let $\mathbf{N}_{i,j} = \mathbf{N}_{i,j} + (\mathbf{T}_{left})_{i,i_q}\cdot x_q\cdot\mathbf{R}_{j_q,j}$.
    (d) For all $j = 1, ..., \xi_2$, let $\mathbf{D}_{i_q,j} = \mathbf{M}_{i_q,j} + x_q\cdot\mathbf{R}_{j_q,j}$
5. end
6. Construct $\mathbf{X}_* = \underset{\mathrm{rank}(\mathbf{X})\le k}{\operatorname{argmin}} \|\mathbf{N}\cdot\mathbf{X}\cdot\mathbf{L}-\mathbf{M}\|_{\mathrm{F}}^2$. (Notice that $\mathbf{X}_* = \underset{\mathrm{rank}(\mathbf{X})\le k}{\operatorname{argmin}} \|\mathbf{T}_{left}(\mathbf{ARXSA}-\mathbf{A})\mathbf{T}_{right}\|_{\mathrm{F}}^2$.)
7. Compute the SVD of $\mathbf{X}_* = \mathbf{U}_{\mathbf{X}_*}\mathbf{\Sigma}_{\mathbf{X}_*}\mathbf{V}_{\mathbf{X}_*}^{\mathrm{T}}$ ($\mathbf{U}_{\mathbf{X}_*} \in \mathbb{R}^{\xi\times k}$, $\mathbf{\Sigma}_{\mathbf{X}_*} \in \mathbb{R}^{k\times k}$ $\mathbf{V}_{\mathbf{X}_*} \in \mathbb{R}^{\xi\times k}$); then, compute $\mathbf{T} = \mathbf{D}\mathbf{U}_{\mathbf{X}_*}$.
8. Compute an orthonormal basis $\mathbf{U} \in \mathbb{R}^{m\times k}$ for $span(\mathbf{T})$.
```

THEOREM 9. *The matrix* $\mathbf{U} \in \mathbb{R}^{m\times k}$ *with* k *orthonormal columns satisfies w.p.* 0.96*:* $\|\mathbf{A}-\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \le (1+O(\varepsilon))\cdot\|\mathbf{A}-\mathbf{A}_k\|_{\mathrm{F}}^2$*. The space complexity is* $O\left(mk/\varepsilon + \mathrm{poly}(k\varepsilon^{-1})\right)$ *machine words, and the running time for each update operation is* $O(poly(k\varepsilon^{-1}))$*. Furthermore, the total running time of the algorithm is of the order* $O\left(l\cdot\mathrm{poly}(k\varepsilon^{-1}) + mk^2\varepsilon^{-1}\right)$ *where* l *is the total number of updates.*

6.1.2 A Variation which Outputs $\mathbf{A}_k^*$

We just slightly modify the previous algorithm in Section 6.1.1 to get the algorithm which can output a factorization of a matrix $\mathbf{A}_k^* \in \mathbb{R}^{m\times n}$ satisfying $\|\mathbf{A}-\mathbf{A}_k^*\|_{\mathrm{F}}^2 \le (1+\varepsilon)\cdot\|\mathbf{A}-\mathbf{A}_k\|_{\mathrm{F}}^2$. All we need to do is to maintain a matrix $\mathbf{C} = \mathbf{SA}$. To achieve this, we only need to update $\mathbf{C}$ in each iteration of step 4 of the algorithm in Section 6.1.1. Additionally, in step 7, we compute a matrix $\mathbf{K} = \mathbf{V}_{\mathbf{X}_*}^{\mathrm{T}}\mathbf{C}$. Finally, we output $\mathbf{A}_k^* = \mathbf{T}\mathbf{\Sigma}_{\mathbf{X}_*}\mathbf{K}$.

THEOREM 10. *With probability at least* 0.96*:* $\|\mathbf{A}-\mathbf{A}_k^*\|_{\mathrm{F}}^2 \le (1+O(\varepsilon))\cdot\|\mathbf{A}-\mathbf{A}_k\|_{\mathrm{F}}^2$*. The space complexity of the algorithm is* $O\left((m+n)k/\varepsilon + \mathrm{poly}(k\varepsilon^{-1})\right)$ *words, and the running time for each update operation is* $O(poly(k\varepsilon^{-1}))$ *and the total running time is of the order* $O\left(l\cdot\mathrm{poly}(k\varepsilon^{-1}) + (m+n)k^2\varepsilon^{-1} + mnk\right)$ *where* l *is total number of updates.*

6.2 Two-pass Streaming PCA

We use the same idea as in the case of the distributed PCA algorithm in Section 5. This leads to a two-pass streaming algorithm for PCA. Again, as in the distributed case, we need to test if the rank of $\mathbf{A}$ is less than $2k$, and then we use one approach if $\mathrm{rank}(\mathbf{A}) < 2k$ and another approach if $\mathrm{rank}(\mathbf{A}) \ge 2k$. Both of these approaches can be implemented with two passes. In the overall streaming PCA algorithm that we would like to design, we can not wait for the algorithm that tests the rank to finish and then start running one of the two PCA protocols, because this will lead to a three-pass algorithm (one pass to test the rank and two passes for the actual PCA protocol). To keep the number of passes to two, we just start running the PCA protocols in parallel with the protocol that tests the rank of the input matrix. In the end of the first pass over the stream of update operations, we already know which protocol to follow, and we just do this, disregarding the other protocol.

We already discussed the streaming version of the algorithm that tests the rank of the matrix in Lemma 3. Below, we describe separately the two streaming PCA protocols.

6.2.1 Streaming PCA Protocol when $\mathrm{rank}(\mathbf{A}) \le 2k$

The idea here is to implement a streaming version of the distributed PCA protocol in Lemma 5.

We first construct the matrix $\mathbf{H}''$ and two affine embedding matrices $\mathbf{T}_{left}$ and $\mathbf{T}_{right}$ as in Lemma 5. In the first pass over the stream, we maintain a matrix $\mathbf{C} = \mathbf{AH}''$. In the second pass, we maintain matrices $\mathbf{T}_{left}\mathbf{C}$, $\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{T}_{right}$ and $\mathbf{T}_{left}\mathbf{A}\mathbf{T}_{right}$. Finally, we can solve the optimization problem: $\mathbf{X}_* = \underset{\mathrm{rank}(\mathbf{X})\le k}{\operatorname{argmin}} \|\mathbf{T}_{left}\mathbf{CXC}^{\mathrm{T}}\mathbf{A}\mathbf{T}_{right} - \mathbf{T}_{left}\mathbf{A}\mathbf{T}_{right}\|_{\mathrm{F}}^2$, and output an orthonormal basis $\mathbf{U}$ for $span(\mathbf{C}\mathbf{X}_*)$.

6.2.2 Streaming PCA Protocol when $\mathrm{rank}(\mathbf{A}) > 2k$

The idea here is to implement a streaming version of step 3 of the algorithm in Section 5.4. We first construct sketching matrices $\mathbf{S}$ and $\mathbf{T}$ as in the algorithm in Section 5.4. In the first pass over the stream, we maintain a matrix $\tilde{\mathbf{B}} = \mathbf{SBT}$, where $\mathbf{B}$ arises from $\mathbf{A}$ after applying the perturbation. After the first pass, we compute the SVD of $\tilde{\mathbf{B}}_k = \mathbf{U}_{\tilde{\mathbf{B}}_k}\mathbf{\Sigma}_{\tilde{\mathbf{B}}_k}\mathbf{V}_{\tilde{\mathbf{B}}_k}^{\mathrm{T}}$. Then, we round each of the entries of $\mathbf{V}_{\tilde{\mathbf{B}}_k}$ to the nearest integer multiple of $1/n^\gamma$ for a sufficient large constant $\gamma > 0$. Let the matrix after the rounding be $\hat{\mathbf{V}}_{\tilde{\mathbf{B}}_k}$. In the second pass, we maintain a matrix $\hat{\mathbf{X}} = \mathbf{BT}\hat{\mathbf{V}}_{\tilde{\mathbf{B}}_k}$. We finally output an orthonormal basis $\mathbf{U}$ for $span(\hat{\mathbf{X}})$.

6.2.3 Main Result

The theorem below analyzes the approximation error, the space, and the running time of the previous algorithm.

THEOREM 11. *The matrix* $\mathbf{U} \in \mathbb{R}^{m\times k}$ *with* k *orthonormal columns satisfies with arbitrarily large constant probability:* $\|\mathbf{A}-\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \le (1+\varepsilon)\cdot\|\mathbf{A}-\mathbf{A}_k\|_{\mathrm{F}}^2$*. The space complexity of the algorithm is* $O\left(mk + \mathrm{poly}(k\varepsilon^{-1})\right)$ *machine words, the running time of each update operation is* $O(poly(k\varepsilon^{-1}))$*, and the total running time is of the order* $O\left(l\cdot\mathrm{poly}(k\varepsilon^{-1}) + mk^2\right)$ *where* l *is the total number of update operations.*

7. PCA FOR SPARSE MATRICES IN COLUMN-PARTITION MODEL

Recall that in the problem of Definition 2 we are given 1) an $m\times n$ matrix $\mathbf{A}$ partitioned column-wise as follows: $\mathbf{A} =$

$\begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2 & \dots & \mathbf{A}_s\end{pmatrix}$, with $\mathbf{A}_i \in \mathbb{R}^{m \times w_i}$ ($\sum w_i = n$); 2) a rank parameter $k < \text{rank}(\mathbf{A})$; 3) an accuracy parameter $\varepsilon > 0$. We would like to design an algorithm that finds an $m \times k$ matrix $\mathbf{U}$ with $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_k$ and, upon termination, leaves this matrix $\mathbf{U}$ in each machine of the network.

The high level idea of our algorithm is to find, in a distributed way, a small set - $O(k\varepsilon^{-1})$ - of columns from $\mathbf{A}$ and then find $\mathbf{U}$ in the span of those columns. To choose those $O(k\varepsilon^{-1})$ columns of $\mathbf{A}$ in a distributed way we implement the following three-stage sampling procedure:

1. Local sampling: Each machine samples $O(k)$ columns using a deterministic algorithm from [13].

2. Global sampling: The server collects the columns from each machine and down-samples them to $O(k)$ columns using the same deterministic algorithm from [13].

3. Adaptive sampling: the server sends back to each machine those $O(k)$ columns; then, an extra of $O(k\varepsilon^{-1})$ columns are selected randomly from the entire matrix $\mathbf{A}$ using [22].

We argue that if $\tilde{\mathbf{C}}$ contains the columns selected with this three-stage approach, then, w.p. .99,

$$\|\mathbf{A}-\tilde{\mathbf{C}}\tilde{\mathbf{C}}^{\dagger}\mathbf{A}\|_{\mathrm{F}}^2 \le \|\mathbf{A}-\Pi_{\tilde{\mathbf{C}},k}^{\mathrm{F}}(\mathbf{A})\|_{\mathrm{F}}^2 \le (1+O(\varepsilon))\cdot\|\mathbf{A}-\mathbf{A}_k\|_{\mathrm{F}}^2.$$

Though we could have used $\Pi_{\tilde{\mathbf{C}},k}^{\mathrm{F}}(\mathbf{A})$ to be the rank k matrix that approximates $\mathbf{A}$, we are not familiar with any communication efficient computation of $\Pi_{\tilde{\mathbf{C}},k}^{\mathrm{F}}(\mathbf{A})$. To address this issue, using an idea from [30], we compute $\mathbf{U} \in span(\tilde{\mathbf{C}})$ such that

$$\|\mathbf{A}-\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|_{\mathrm{F}}^2 \le (1+O(\varepsilon))\|\mathbf{A}-\Pi_{\tilde{\mathbf{C}},k}^{\mathrm{F}}(\mathbf{A})\|_{\mathrm{F}}^2;$$

this $\mathbf{U}$ can be calculated with small communication cost and it is sufficient for our purposes.

Before presenting the new algorithm in detail, we discuss results from previous literature.

7.1 Background Material

7.1.1 Column Sampling Algorithms

Our distributed PCA algorithm in Section 7 samples columns from the input matrix in three stages. In the first two stages, we use a deterministic algorithm developed in [13], which itself extends the Batson, Spielman, and Strivastava (BSS) spectral sparsification algorithm [7]. For the actual algorithm we defer the reader to Lemma 3.6 in [13]. Lemma 8 and Lemma 9 below present the relevant results. In the third sampling stage, we use an adaptive sampling algorithm from [22].

Lemma 8. *(Lemma 3.6 in [13]) Let $\mathbf{V} \in \mathbb{R}^{w \times k}$ be a matrix with $w > k$ and $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}_k$. Let $\mathbf{E} \in \mathbb{R}^{m \times w}$ be an arbitrary matrix. Then, given an integer ℓ such that $k < \ell \le w$, there exists a deterministic algorithm that runs in $O\left(\ell w k^2 + m w\right)$ time, and constructs a "sampling/rescaling" $w \times \ell$ matrix $\mathbf{S}$ such that*

$$\sigma_k^2\left(\mathbf{V}^{\mathrm{T}}\mathbf{S}\right) \ge \left(1-\sqrt{k/\ell}\right)^2, \qquad \|\mathbf{E}\mathbf{S}\|_{\mathrm{F}}^2 \le \|\mathbf{E}\|_{\mathrm{F}}^2.$$

Specifically, $rank(\mathbf{V}^{\mathrm{T}}\mathbf{S}) = k$. We denote this sampling procedure as $\mathbf{S} = BssSampling(\mathbf{V}, \mathbf{E}, \ell)$.

Lemma 9. *(Theorem 5 in [13]) Given matrix $\mathbf{G} \in \mathbb{R}^{m \times \alpha}$ of rank ρ and a target rank k [1], there exists a deterministic algorithm that runs in $O\left(\alpha m \min\{\alpha, m\} + \alpha c k^2\right)$ time and selects $c > k$ columns of $\mathbf{G}$ to form a matrix $\mathbf{C} \in \mathbb{R}^{m \times c}$ with*

$$\|\mathbf{G}-\mathbf{C}\mathbf{C}^{\dagger}\mathbf{G}\|_{\mathrm{F}}^2 \le \left(1+\left(1-\sqrt{k/c}\right)^{-2}\right)\cdot\sum_{i=k+1}^{rank(\mathbf{G})}\sigma_i^2(\mathbf{G}).$$

The algorithm in this theorem finds $\mathbf{C}$ as $\mathbf{C} = \mathbf{G}\mathbf{S}$, where $\mathbf{S} = BssSampling(\mathbf{V}, \mathbf{G}-\mathbf{G}\mathbf{V}\mathbf{V}^{\mathrm{T}}, c)$ and $\mathbf{V} \in \mathbb{R}^{\alpha \times k}$ contains the top k right singular vectors of $\mathbf{G}$. We denote this sampling procedure as $\mathbf{C} = DeterministicCssFrobenius(\mathbf{G}, k, c)$.

Lemma 10. *(Adaptive sampling; Theorem 2.1 of [22]) Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{V} \in \mathbb{R}^{m \times c_1}$ (with $c_1 \le n, m$), define the residual matrix $\mathbf{\Psi} = \mathbf{A} - \mathbf{V}\mathbf{V}^{\dagger}\mathbf{A} \in \mathbb{R}^{m \times n}$. For $j = 1, \dots, n$, let p_j be a probability distribution such that $p_j \ge \beta\|\mathbf{\Psi}^{(j)}\|_2^2/\|\mathbf{\Psi}\|_{\mathrm{F}}^2$, for some $1 > \beta > 0$, where $\mathbf{\Psi}^{(j)}$ is the j-th column of the matrix $\mathbf{\Psi}$. Sample c_2 columns from $\mathbf{A}$ in c_2 i.i.d. trials, where in each trial the j-th column is chosen with probability p_j. Let $\mathbf{C}_2 \in \mathbb{R}^{m \times c_2}$ contain the c_2 sampled columns and let $\mathbf{C} = [\mathbf{V} \;\; \mathbf{C}_2] \in \mathbb{R}^{m \times (c_1+c_2)}$ contain the columns of $\mathbf{V}$ and $\mathbf{C}_2$. Then, for any integer $k > 0$, $\mathbb{E}\left[\|\mathbf{A}-\mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}\|_{\mathrm{F}}^2\right] \le \mathbb{E}\left[\|\mathbf{A}-\Pi_{\mathbf{C},k}^{\mathrm{F}}(\mathbf{A})\|_{\mathrm{F}}^2\right]$*

$$\le \sum_{i=k+1}^{rank(\mathbf{A})}\sigma_i^2(\mathbf{A}) + \frac{k}{\beta \cdot c_2}\cdot\|\mathbf{A}-\mathbf{V}\mathbf{V}^{\dagger}\mathbf{A}\|_{\mathrm{F}}^2.$$

Given $\mathbf{A}$ and $\mathbf{C}$, this method requires $O(c_1 m n)$ time to compute $\mathbf{\Psi}$, another $O(mn)$ time to compute the probabilities p_j's and another $O(n + c_2)$ time for the sampling step - using the method in [43]. In total, the method requires $O(c_1 mn + c_2)$ time to compute $\mathbf{C}_2$. We denote this as $\mathbf{C}_2 = AdaptiveCols(\mathbf{A}, \mathbf{V}, c_2, \beta)$.

7.1.2 Distributed Adaptive Sampling

In our distributed PCA algorithm below, we also need to use a distributed version of the previous adaptive sampling procedure. We provide some preliminary results for that task in this section.

Lemma 11. *Suppose that the columns of $\mathbf{A} \in \mathbb{R}^{m \times n}$ are partitioned arbitrarily across the machines into matrices $\mathbf{A}_1, \dots, \mathbf{A}_s$. Let $\mathbf{C}$ be an arbitrary $m \times r$ matrix. Consider the distribution p on n columns in which $p_j = \frac{\|\mathbf{a}_j-\mathbf{C}\mathbf{C}^{\dagger}\mathbf{a}_j\|_{\mathrm{F}}^2}{\|\mathbf{A}-\mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}\|_{\mathrm{F}}^2}$, where $\mathbf{a}_j$ is the j-th column of $\mathbf{A}$ ($j = 1 : n$ here).*

For each $i \in [s]$, let some value β_i satisfies $\|\mathbf{A}_i - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}_i\|_{\mathrm{F}}^2 \le \beta^i \le 2\|\mathbf{A}_i - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}_i\|_{\mathrm{F}}^2$. For each $j \in [n]$, if column $\mathbf{a}_j$ is held on the i-th server (denoted $\mathbf{a}_j^i$), then let $q_j = \frac{\beta^i}{\sum_{i'=1}^{s}\beta_{i'}} \cdot \frac{\|\mathbf{a}_j^i-\mathbf{C}\mathbf{C}^{\dagger}\mathbf{a}_j^i\|_2^2}{\|\mathbf{A}_i-\mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}_i\|_{\mathrm{F}}^2}$. Then for each $j \in [n]$, $p_j/2 \le q_j \le 2p_j$.

Lemma 12. *$O(\log k + \log\log(mns/\varepsilon))$ Suppose the coordinator has an $m \times r$ matrix $\mathbf{C}$ of columns of an $m \times n$ matrix $\mathbf{A}$, where $r = O(k)$. Suppose the entries of $\mathbf{A}$ are integers bounded by $\mathrm{poly}(mns/\varepsilon)$ in magnitude, and let the columns of $\mathbf{A}$ be partitioned arbitrarily across the servers into matrices $\mathbf{A}_1, \dots, \mathbf{A}_s$.*

There is a protocol using $O(skm)$ machine words of communication for the coordinator to learn values β^i so that for all $i \in [s]$, $\|\mathbf{A}_i - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}_i\|_{\mathrm{F}}^2 \le \beta^i \le 2\|\mathbf{A}_i - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}_i\|_{\mathrm{F}}^2$.

[1] The original Theorem 5 in [13] has the assumption that $k < \rho$, but this assumption can be dropped having the result unchanged. The only reason the assumption $k < \rho$ exists is because otherwise column subset selection is trivial.

7.1.3 Low-rank Matrix Approximations within a Subspace

The final stage of our distributed PCA algorithm below finds $\mathbf{U} \in span(\tilde{\mathbf{C}})$ such that the error of the residual matrix $\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}$ is "small". To implement this step, we employ an algorithm developed in [30].

LEMMA 13. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be the input matrix and $\mathbf{V} \in \mathbb{R}^{m \times c}$ be the input subspace. We further assume that for some rank parameter $k < c$ and accuracy parameter $0 < \varepsilon < 1$:*

$$\|\mathbf{A} - \Pi^{\mathrm{F}}_{\mathbf{V},k}(\mathbf{A})\|^2_{\mathrm{F}} \leq (1 + O(\epsilon))\|\mathbf{A} - \mathbf{A}_k\|^2_{\mathrm{F}}.$$

Let $\mathbf{V} = \mathbf{Y}\mathbf{\Psi}$ be a qr decomposition of $\mathbf{V}$ with $\mathbf{Y} \in \mathbb{R}^{m \times c}$ and $\mathbf{\Psi} \in \mathbb{R}^{c \times c}$. Let $\mathbf{\Xi} = \mathbf{Y}^{\mathrm{T}}\mathbf{A}\mathbf{W}^{\mathrm{T}} \in \mathbb{R}^{c \times \xi}$, where $\mathbf{W}^{\mathrm{T}} \in \mathbb{R}^{n \times \xi}$ with $\xi = O(c/\varepsilon^2)$, each element of which is chosen i.i.d. to be $\{+1/\sqrt{n}, -1/\sqrt{n}\}$ with probability $1/2$. Let $\mathbf{\Delta} \in \mathbb{R}^{c \times k}$ contain the top k left singular vectors of $\mathbf{\Xi}$. Then, with probability at least $1 - e^{-c}$, $\|\mathbf{A} - \mathbf{Y}\mathbf{\Delta}\mathbf{\Delta}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{A}\|^2_{\mathrm{F}} \leq (1 + \varepsilon)\|\mathbf{A} - \mathbf{A}_k\|^2_{\mathrm{F}}$. $\mathbf{Y}$, and $\mathbf{\Delta}$ can be computed in $O(mn\xi)$ time. We denote this procedure as $[\mathbf{Y}, \mathbf{\Delta}] = ApproxSubspaceSVD(\mathbf{A}, \mathbf{V}, k, \varepsilon)$.

7.2 Detailed Description of the Algorithm

Input:

1. $\mathbf{A} \in \mathbb{R}^{m \times n}$ partitioned column-wise $\mathbf{A} = \begin{pmatrix}\mathbf{A}_1 & \dots & \mathbf{A}_s\end{pmatrix}$; for $i = 1 : s$, $\mathbf{A}_i \in \mathbb{R}^{m \times w_i}$; $\sum_i w_i = n$.
2. rank parameter $k < \mathrm{rank}(\mathbf{A})$
3. accuracy parameter $\varepsilon > 0$

Algorithm

1. **Local Column Sampling**
   (a) For each sub-matrix $\mathbf{A}_i \in \mathbb{R}^{m \times w_i}$, compute the matrix $\mathbf{V}_i \in \mathbb{R}^{w_i \times k}$ with the top k right singular vectors of $\mathbf{A}_i$. Also, construct the $m \times w_i$ matrix $\mathbf{E}_i = \mathbf{A}_i - \mathbf{A}_i\mathbf{V}_i\mathbf{V}_i^{\mathrm{T}}$. For each sub-matrix $\mathbf{A}_i$, compute $\mathbf{C}_i \in \mathbb{R}^{m \times \ell}$ containing $\ell = 4k$ columns from $\mathbf{A}_i$ as follows: $\mathbf{C}_i = \mathbf{A}_i\mathbf{S}_i$. Here, $\mathbf{S}_i$ has dimensions $w_i \times \ell$ and is constructed as follows: $\mathbf{S}_i = BssSampling(\mathbf{V}_i, \mathbf{E}_i, \ell)$ (see Lemma 8).
   (b) Machine i sends $\mathbf{C}_i$ to the server.
2. **Global Column Sampling**
   (a) Server constructs $m \times (s \cdot \ell)$ matrix $\mathbf{G}$ containing $(s \cdot \ell)$ actual columns from $\mathbf{A}$ as: $\mathbf{G} = \begin{pmatrix}\mathbf{C}_1 & \mathbf{C}_2 & \dots & \mathbf{C}_s\end{pmatrix}$. Server uses $\mathbf{C} = DeterministicCssFrobenius(\mathbf{G}, k, c_1)$ to construct $\mathbf{C} \in \mathbb{R}^{m \times c_1}$ via choosing $c_1 = 4k$ columns from $\mathbf{G}$ (see Lemma 9).
   (b) Server sends $\mathbf{C}$ to all the machines.
3. **Adaptive Column Sampling**
   (a) Machine i computes $\mathbf{\Psi}_i = \mathbf{A}_i - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}_i \in \mathbb{R}^{m \times w_i}$ and then computes β_i as it was described in Lemma 12. Machine i sends β_i to server.
   (b) Server computes probability distribution $g_i = \frac{\beta_i}{\sum_i \beta_i}$. Server samples i.i.d. with replacement $\lceil 50k/\varepsilon \rceil$ samples (machines) from g_i. Then, server determines numbers t_i ($i = 1, 2, \dots, s$), where t_i is the number of times the ith machine was sampled. It sends the t_i's to the machines.
   (c) Machine i computes probabilities $q^i_j = \|\mathbf{x}\|^2_2/\|\mathbf{\Psi}_i\|^2_{\mathrm{F}}$ ($j = 1 : w_i$), where $\mathbf{x}$ is the jth column of $\mathbf{\Psi}_i$. And now machine i samples t_i samples from it's local probability distribution and sends the corresponding columns to the server. Let $c_2 = \sum_i t_i = \lceil 50k/\varepsilon \rceil$.
   (d) Server collects the columns and assigns them to $\hat{\mathbf{C}} \in \mathbb{R}^{m \times c_2}$. Server constructs $\tilde{\mathbf{C}}$ to be the $m \times (c_1 + c_2)$ matrix: $\tilde{\mathbf{C}} = \begin{pmatrix}\mathbf{C}; & \hat{\mathbf{C}}\end{pmatrix}$. Let $c = c_1 + c_2 = 4k + \lceil 50k/\varepsilon \rceil$.
4. **Rank-k matrix in the span of $\tilde{\mathbf{C}}$**
   (a) Server sends $\tilde{\mathbf{C}}$ to all the machines and each machine computes (the same) qr factorization of $\tilde{\mathbf{C}}$: $\tilde{\mathbf{C}} = \mathbf{Y}\mathbf{R}$ where $\mathbf{Y} \in \mathbb{R}^{m \times c}$ has orthonormal columns and $\mathbf{R} \in \mathbb{R}^{c \times c}$ is upper triangular.
   (b) Machine i generates $\tilde{\mathbf{W}}_i \in \mathbb{R}^{\xi \times w_i}$ with $\xi = O(c/\varepsilon^2)$ to be i.i.d. $\{+1, -1\}$ w.p. $1/2$. *Implicitly* all machines together generate $\tilde{\mathbf{W}} = \begin{pmatrix}\tilde{\mathbf{W}}_1 & \tilde{\mathbf{W}}_2 & \dots & \tilde{\mathbf{W}}_s\end{pmatrix}$, with $\tilde{\mathbf{W}} \in \mathbb{R}^{\xi \times n}$. Machine i computes $\mathbf{H}_i = \tilde{\mathbf{C}}^{\mathrm{T}}(\mathbf{A}_i\mathbf{W}_i^{\mathrm{T}}) \in \mathbb{R}^{c \times \xi}$. Machine i sends $\mathbf{H}_i$ to the server.
   (c) Server computes $\mathbf{\Xi} = \sum_{i=1}^{s} \mathbf{H}_i \in \mathbb{R}^{c \times \xi}$ and sends this back to all the machines. Now machines compute $\mathbf{\Xi} := \mathbf{R}^{-1} \cdot \mathbf{\Xi} \cdot 1/\sqrt{n} (:= \mathbf{Y}^{\mathrm{T}}\mathbf{A}\mathbf{W}^{\mathrm{T}})$, where the matrix $\mathbf{W}$ satisfies $\mathbf{W} = \begin{pmatrix}\mathbf{W}_1 & \mathbf{W}_2 & \dots & \mathbf{W}_s\end{pmatrix} \in \mathbb{R}^{\xi \times n}$ is a random matrix each element of which is taking values $\{+1/\sqrt{n}, -1/\sqrt{n}\}$ w.p. $1/2$, and then they compute $\mathbf{\Delta} \in \mathbb{R}^{c \times k}$ to be the top k left singular vectors of $\mathbf{\Xi}$. Each machine computes $\mathbf{U} = \mathbf{Y} \cdot \mathbf{\Delta} \in \mathbb{R}^{m \times k}$.

Discussion.

A few remarks are necessary for the last two stages of the algorithm. The third stage (adaptive column sampling), implements the adaptive sampling method of Lemma 10. To see this, note that each column in $\mathbf{A}$ (specifically the jth column in $\mathbf{A}_i$) is sampled with probability $g_i \cdot q^i_j \geq \frac{1}{2} \cdot \frac{||\mathbf{x}^i_j||^2_2}{\|\mathbf{\Psi}\|^2_{\mathrm{F}}}$, where $\mathbf{\Psi} = \mathbf{A} - \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}$. This follows from Lemma 11. Overall, this stage effectively constructs $\hat{\mathbf{C}}$ such that (see Lemma 10) $\hat{\mathbf{C}} = AdaptiveCols(\mathbf{A}, \mathbf{C}, c_2, 1/2)$. The last stage in the algorithm implements the algorithm in Lemma 13. To see this, note that $\mathbf{W}$ satisfies the properties in the lemma. Hence, $[\mathbf{Y}, \mathbf{\Delta}] = ApproxSubspaceSVD(\mathbf{A}, \tilde{\mathbf{C}}, k, \varepsilon)$.

7.3 Main Result

THEOREM 12. *The matrix $\tilde{\mathbf{C}} \in \mathbb{R}^{m \times c}$ with $c = O(k/\varepsilon)$ columns of $\mathbf{A}$ satisfies w.p. at least $.99$,*

$$\|\mathbf{A} - \tilde{\mathbf{C}}\tilde{\mathbf{C}}^{\dagger}\mathbf{A}\|^2_{\mathrm{F}} \leq \|\mathbf{A} - \Pi^{\mathrm{F}}_{\tilde{\mathbf{C}},k}(\mathbf{A})\|^2_{\mathrm{F}} \leq (1 + 200\varepsilon)\|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^{\mathrm{T}}\mathbf{A}\|^2_{\mathrm{F}}. \quad (4)$$

The matrix $\mathbf{U} \in \mathbb{R}^{m \times k}$ with k orthonormal columns satisfies with probability at least $.98$,

$$\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|^2_{\mathrm{F}} \leq (1 + \varepsilon) \cdot \|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^{\mathrm{T}}\mathbf{A}\|^2_{\mathrm{F}}. \quad (5)$$

Let each column of $\mathbf{A}$ have at most ϕ non-zero elements. Then, the communication cost of the algorithm is $O\left(sk\phi\varepsilon^{-1} + sk^2\varepsilon^{-4}\right)$ machine words and the running time is $O(mn \min\{m, n\} + mns \cdot poly(k, 1/\varepsilon))$.

8. FASTER DISTRIBUTED PCA FOR SPARSE MATRICES

One can modify certain steps of the distributed PCA algorithm in Section 7 to improve the total running time by using input sparsity sketches and related techniques in [18, 15].

THEOREM 13. *The matrix $\tilde{\mathbf{C}} \in \mathbb{R}^{m \times c}$ with $c = O(k/\varepsilon)$ columns of $\mathbf{A}$ satisfies w.p. $0.59 - \frac{s+1}{n} - 2\delta$:*

$$\|\mathbf{A} - \tilde{\mathbf{C}}\tilde{\mathbf{C}}^{\dagger}\mathbf{A}\|^2_{\mathrm{F}} \leq \|\mathbf{A} - \Pi^{\mathrm{F}}_{\tilde{\mathbf{C}},k}(\mathbf{A})\|^2_{\mathrm{F}} \leq (1 + O(\varepsilon)) \cdot \|\mathbf{A} - \mathbf{A}_k\|^2_{\mathrm{F}}.$$

The matrix $\mathbf{U} \in \mathbb{R}^{m \times k}$ with k orthonormal columns satisfies w.p. $0.58 - \frac{s+1}{n} - 2\delta$: $\|\mathbf{A} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{A}\|^2_{\mathrm{F}} \leq (1 + \varepsilon) \cdot \|\mathbf{A} - \mathbf{A}_k\|^2_{\mathrm{F}}$. Let each column of $\mathbf{A}$ have at most ϕ non-zero elements. Then, the communication is $O\left(sk\phi\varepsilon^{-1} + sk^3\varepsilon^{-5}\right)$ words and the time is $O\left(\mathrm{nnz}(\mathbf{A}) \cdot \log^2\left(\frac{ns}{\delta}\right) + (m + n)s \cdot poly(k, \varepsilon^{-1}, \log\left(\frac{ns}{\delta}\right)\right)$.

9. LOWER BOUNDS

9.1 Lower Bounds for Distributed PCA on Dense Matrices

This section provides communication cost lower bounds for the Distributed PCA problem of Definition 4.

9.1.1 *Preliminaries*

We use the notation $\mathbf{G}_{k,m}$ to denote the set of k-dimensional subspaces of $\mathbb{R}^m$, which we identify with corresponding projector matrices $\mathbf{Q} \in \mathbb{R}^{m\times m}$, with $\mathrm{rank}(\mathbf{Q}) = k \le m$, onto the subspaces, i.e., $\mathbf{G}_{k,m} = \{\mathbf{Q} \in \mathbb{R}^{m\times m} : \mathbf{Q}^2 = \mathbf{Q}, \mathrm{rank}(\mathbf{Q}) = k \le m.\}$ We also need a description of a subset from $\mathbf{G}_{k,m}$: $C^k_\delta(\mathbf{P}) = \{\mathbf{Q} \in \mathbf{G}_{k,m} : \|\mathbf{P}-\mathbf{Q}\|_2 \le \delta\}$.

THEOREM 14. *(Net Bound - Corollary 5.1 of [31]) For any $m, k, \delta > 0$, there is a family $\mathcal{N} = \{\mathbf{P}^1, \dots, \mathbf{P}^N\}$ and $N = 2^{\Omega(k(m-k)\log(1/\delta))}$, where $\mathbf{P}^i \in \mathbf{G}_{k,m}$ and $C^k_\delta(\mathbf{P}^i) \cap C^k_\delta(\mathbf{P}^j) = \emptyset$, for all $i \neq j$.*

9.1.2 *Hard Instance Construction*

Below, we describe a specific construction for a matrix $\mathbf{A}$. First of all, we set $\delta = 1/(skm)$, the parameter to be used in Theorem 14. Next, fix a family $\mathcal{N}$ of subspaces of $\mathbf{G}_{k,m}$ with the guarantees of Theorem 14. Let $\mathbf{P}_\mathbf{R} = \mathbf{R}\mathbf{R}^\mathrm{T}$ be a uniformly random member of $\mathcal{N}$, where $\mathbf{R} \in \mathbb{R}^{m\times k}$ has orthonormal columns. The entries of $\mathbf{A}_1$ are equal to the entries of $\mathbf{R}$, each rounded to the nearest integer multiple of $1/B$, where $B = \mathrm{poly}(skm)$ is a large enough parameter specified later. We denote this rounded matrix with $\tilde{\mathbf{R}} \in \mathbb{R}^{m\times k}$. Each $\mathbf{A}_i$ for $i = 2, 3, \dots, s-1$ is $1/B$ times the $m \times m$ identity matrix. Finally, $\mathbf{A}_s$ is the $m \times t$ matrix of all zeros with $t = n - (s-1)m - k$. I.e., $\mathbf{A} = \begin{pmatrix}\tilde{\mathbf{R}} & \frac{1}{B}\mathbf{I}_m & \frac{1}{B}\mathbf{I}_m & \dots & \frac{1}{B}\mathbf{I}_m & \mathbf{0}_{m\times t}\end{pmatrix}$.

THEOREM 15. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ be the hard instance matrix described above and be distributed in s machines in the above way. Suppose $n = \Omega(sm)$, $k \le .99m$, and $1 < C < poly(skm)$. Assume that there is an algorithm (protocol) which succeeds with probability at least $2/3$ in having the i-th server output $\mathbf{Q}\mathbf{A}_i$, for all i, where $\mathbf{Q} \in \mathbb{R}^{m\times m}$ is a projector matrix with rank at most k such that $\|\mathbf{A} - \mathbf{Q}\mathbf{A}\|_\mathrm{F} \le C\|\mathbf{A} - \mathbf{A}_k\|_\mathrm{F}$. Then, this algorithm requires $\Omega(skm\log(skm))$ bits of communication.*

Further, the bound holds even if the input matrices $\mathbf{A}_i$ to the servers have all entries which are integer multiples of $1/B$, for a value $B = poly(skm)$, and bounded in magnitude by 1, and therefore all entries can be specified with $O(\log(skm))$ bits.

Note that assuming a word size of $\Theta(\log(skm))$ bits, we obtain an $\Omega(skm)$ word lower bound.

9.2 Lower Bounds for Distributed PCA on Sparse Matrices

COROLLARY 2. *Let $\mathbf{A} \in \mathbb{R}^{\phi\times n}$ be the hard instance matrix in Section 9.1.2 (with m replaced with ϕ in noting the number of rows in $\mathbf{A}$): $\mathbf{A} = \begin{pmatrix}\tilde{\mathbf{R}} & \frac{1}{B}\mathbf{I}_\phi & \dots & \frac{1}{B}\mathbf{I}_\phi & \mathbf{0}_{\phi\times t}\end{pmatrix}$. Suppose $n = \Omega(s\phi)$, $k \le .99\phi$, and $1 < C < poly(sk\phi)$. Now, for arbitrary m consider the matrix $\hat{\mathbf{A}} \in \mathbb{R}^{m\times n}$ which has $\mathbf{A}$ in the top part and the all-zeros $(m-\phi)\times n$ matrix in the bottom part. $\hat{\mathbf{A}}$ admits the same column partition as $\mathbf{A}$ after padding with zeros:*

$$\hat{\mathbf{A}} = \begin{pmatrix}\tilde{\mathbf{R}} & \frac{1}{B}\mathbf{I}_\phi & \dots & \frac{1}{B}\mathbf{I}_\phi & \mathbf{0}_{\phi\times t} \\ \mathbf{0}_{(m-\phi)\times k} & \mathbf{0}_{(m-\phi)\times\phi} & \dots & \mathbf{0}_{(m-\phi)\times\phi} & \mathbf{0}_{(m-\phi)\times t}\end{pmatrix}.$$

We denote the new sub-matrices with $\hat{\mathbf{A}}_i \in \mathbb{R}^{m\times w_i}$. Assume that there is an algorithm (protocol) which succeeds with probability at least $2/3$ in having the i-th server output $\hat{\mathbf{Q}}\hat{\mathbf{A}}_i$, for all i, where $\hat{\mathbf{Q}} \in \mathbb{R}^{m\times m}$ is a projector matrix with rank at most k such that $\|\hat{\mathbf{A}} - \hat{\mathbf{Q}}\hat{\mathbf{A}}\|_\mathrm{F} \le C\|\hat{\mathbf{A}} - \hat{\mathbf{A}}_k\|_\mathrm{F}$. Then, this algorithm requires $\Omega(sk\phi\log(sk\phi))$ bits of communication.

Note that assuming a word size of $\Theta(\log(sk\phi))$ bits, we obtain an $\Omega(sk\phi)$ word lower bound.

9.3 Lower Bounds for Distributed Column Based Matrix Reconstruction

These lower bounds were outlined a bit in Section 1. Please see the full version [16] for details.

THEOREM 16. *Assume $2/\varepsilon \le \phi$ and $k\phi \le \min(m, n)$. Then, any, possibly randomized, protocol Π which succeeds with probability at least $2/3$ in solving the Distributed Column Subset Selection Problem of Definition 5, under the promise that each column of $\mathbf{A}$ has at most ϕ non-zero entries which are integer multiples of $1/(mn)^c$ and bounded in magnitude by 1, for a constant $c > 0$, requires $\Omega(s\phi k(\log(mn))/\varepsilon)$ bits of communication. If the word size is $\Theta(\log(mn))$, this is $\Omega(s\phi k/\varepsilon)$ words.*

COROLLARY 3. *Assume $2/\varepsilon \le \phi$ and $k\phi \le \min(m, n)$. Then, any, possibly randomized, protocol Π which succeeds with probability at least $2/3$ in solving the Distributed Column Subset Selection Problem - rank k subspace version (see Definition 6), promise that each column of $\mathbf{A}$ has at most ϕ non-zero entries which are integer multiples of $1/(mn)^c$ and bounded in magnitude by 1, for a constant $c > 0$, requires $\Omega(s\phi k(\log(mn))/\varepsilon)$ bits of communication. This is $\Omega(s\phi k/\varepsilon)$ words of size $\Theta(\log(mn))$.*

10. ACKNOWLEDGMENT

D. Woodruff would like to thank support in part from the XDATA program of the Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory FA8750-12-C-0323. P. Zhong was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61361136003.

11. REFERENCES

[1] http://hadoop.apache.org/.

[2] https://mahout.apache.org.

[3] https://spark.apache.org/.

[4] https://spark.apache.org/mllib/.

[5] Z.-J. Bai, R. H. Chan, and F. T. Luk. Principal component analysis for distributed data sets with updating. In *Advanced Parallel Processing Technologies*, pages 471–483. Springer, 2005.

[6] M.-F. Balcan, V. Kanchanapally, Y. Liang, and D. Woodruff. Improved distributed principal component analysis. *arXiv preprint arXiv:1408.5823, to appear in NIPS*, 2014.

[7] J. Batson, D. Spielman, and N. Srivastava. Twice-ramanujan sparsifiers. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 255–262. ACM, 2009.

[8] A. R. Benson, D. F. Gleich, and J. Demmel. Direct qr factorizations for tall-and-skinny matrices in mapreduce architectures. In *Big Data, 2013 IEEE International Conference on*, pages 264–272. IEEE, 2013.

[9] S. Bhojanapalli, P. Jain, and S. Sanghavi. Tighter low-rank approximation via sampling the leveraged element. In *SODA*, pages 902–920. SIAM, 2015.

[10] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, et al. *ScaLAPACK users' guide*, volume 4. siam, 1997.

[11] Y.-A. L. Borgne, S. Raybaud, and G. Bontempi. Distributed principal component analysis for wireless sensor networks. *Sensors*, 2008.

[12] J. Bourgain, S. Dirksen, and J. Nelson. Toward a unified theory of sparse dimensionality reduction in euclidean space. In *STOC*, 2015.

[13] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near optimal column based matrix reconstruction. *SIAM Journal on Computing (SICOMP)*, 2013.

[14] C. Boutsidis and A. Gittens. Improved matrix algorithms via the subsampled randomized hadamard transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1301–1340, 2013.

[15] C. Boutsidis and D. P. Woodruff. Optimal cur matrix decompositions. *STOC*, 2014.

[16] C. Boutsidis, D. P. Woodruff, and P. Zhong. Optimal principal component analysis in distributed and streaming models. *CoRR*, abs/1504.06729, 2015.

[17] K. Clarkson and D. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st annual ACM symposium on Theory of computing (STOC)*, 2009.

[18] K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. In *In STOC*, 2013.

[19] K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. *ArxiV report:* `http://arxiv.org/pdf/1207.6365v4.pdf`, 2013.

[20] M. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for k-means clustering and low rank approximation. *arXiv preprint arXiv:1410.6801*, 2014.

[21] M. B. Cohen. Simpler and tighter analysis of sparse oblivious subspace embeddings. In *SODA*, 2016.

[22] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(12):225–247, 2006.

[23] A. K. Farahat, A. Elgohary, A. Ghodsi, and M. S. Kamel. Distributed column subset selection on mapreduce. In *ICDM*, 2013.

[24] D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *SODA*, pages 1434–1453. SIAM, 2013.

[25] S. Friedland and A. Torokhti. Generalized rank-constrained matrix approximations. *SIAM Journal on Matrix Analysis and Applications*, 29(2):656–659, 2007.

[26] M. Ghashami and J. Phillips. Relative errors for deterministic low-rank matrix approximations. *In SODA*, 2013.

[27] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[28] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal svd. *SIAM Journal on Matrix Analysis and Applications*, 16(1):79–92, 1995.

[29] E. Jessup and D. Sorensen. A parallel algorithm for computing the singular value decomposition of a matrix. *Siam Journal on Matrix Analysis and Applications*, 15(2):530–548, 1994.

[30] R. Kannan, S. S. Vempala, and D. P. Woodruff. Principal component analysis and higher correlations for distributed data. In *Proceedings of The 27th Conference on Learning Theory*, pages 1040–1057, 2014.

[31] M. Kapralov and K. Talwar. On differentially private low rank approximation. In *SODA*, 2013.

[32] D. Kempe and F. McSherry. A decentralized algorithm for spectral analysis. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 561–568. ACM, 2004.

[33] E. Liberty. Simple and deterministic matrix sketching. In *KDD*, pages 581–588. ACM, 2013.

[34] S. V. Macua, P. Belanovic, and S. Zazo. Consensus-based distributed principal component analysis in wireless sensor networks. In *SPAWC*, 2010.

[35] X. Meng and M. W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *STOC*, 2013.

[36] J. Nelson and H. L. Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *FOCS*, 2013.

[37] J. M. Phillips, E. Verbin, and Q. Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *SODA*, 2012.

[38] J. Poulson, B. Marker, R. A. van de Geijn, J. R. Hammond, and N. A. Romero. Elemental: A new framework for distributed memory dense matrix computations. *TOMS*, 39(2), 2013.

[39] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist. Principal component analysis for dimension reduction in massive distributed data sets. In *ICDM*, 2002.

[40] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[41] K. C. Sou and A. Ranzer. On generalized matrix approximation problem in the spectral norm. *Linear Algebra and its Applications*, 436(7):2331–2341, 2012.

[42] F. Tisseur and J. Dongarra. A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures. *SIAM Journal on Scientific Computing*, 20(6), 1999.

[43] M. D. Vose. A linear algorithm for generating random numbers with a given distribution. *Software Engineering, IEEE Transactions on*, 17(9):972–975, 1991.

[44] V. H. Vu and T. Tao. The condition number of a randomly perturbed matrix. In *STOC*, 2007.

[45] D. Woodruff. Low rank approximation lower bounds in row-update streams. In *Advances in Neural Information Processing Systems*, pages 1781–1789, 2014.

[46] D. Woodruff and P. Zhong. Distributed low rank approximation of implicit functions of a matrix. In *ICDE*, 2016.

[47] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.