Iterative Row Sampling

Mu Li
Computer Science Department
CMU
Pittsburgh, USA
muli@cs.cmu.edu

Gary L. Miller
Computer Science Department
CMU
Pittsburgh, USA
glmiller@cs.cmu.edu

Richard Peng
Computer Science Department
CMU
Pittsburgh, USA
yangp@cs.cmu.edu

***Abstract*—There has been significant interest and progress recently in algorithms that solve regression problems involving tall and thin matrices in input sparsity time. Given a $n \times d$ matrix where $n \gg d$, these algorithms find an approximation with fewer rows, allowing one to solve a $\mathrm{poly}(d)$ sized problem instead. In practice, the best performances are often obtained by invoking these routines in an iterative fashion. We show these iterative methods can be adapted to give theoretical guarantees comparable to and better than the current state of the art.**

Our approaches are based on computing the importances of the rows, known as leverage scores, in an iterative manner. We show that alternating between computing a short matrix estimate and finding more accurate approximate leverage scores leads to a series of geometrically smaller instances. This gives an algorithm whose runtime is input sparsity plus an overhead comparable to the cost of solving a regression problem on the smaller approximation. Our results build upon the close connection between randomized matrix algorithms, iterative methods, and graph sparsification.

***Keywords*-Sampling, Regression, Well-conditioned Basis**

I. Introduction

Least squares and ℓ_p regression are among the most common computational linear algebraic operations. In the simplest form, given a matrix $\mathbf{A}$ and a vector $\mathbf{b}$, the regression problem aims to find $\mathbf{x}$ that minimizes:

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_p$$

Where $\|\cdot\|_p$ denotes the ℓ_p-norm of a vector, $\|\mathbf{z}\|_p = (\sum_i |z_i|^p)^{1/p}$. The case of $p = 2$ reduces to the problem of solving a linear system involving the positive semi-definite matrix $\mathbf{A}^T\mathbf{A}$ [9], and is one of the most extensively studied algorithmic questions. Over the past two decades, it was shown that ℓ_1 regression has good properties in recovering structural information [10]. These results make regression algorithms a key tool in data analysis, machine learning, as well as a subroutine in other algorithms.

The ever growing sizes of data raises the natural question of algorithmic efficiency of regression routines. In the most general setting, the answer is far from satisfying with the only general purpose tool being convex optimization. When $\mathbf{A}$ is $n \times d$, the state of the theoretical runtime is about $O((n + d)^{3/2}d)$ [11]. In fact, even in the ℓ_2 case, the best general purpose algorithm takes $O(nd^{\omega-1})$ time where $\omega \approx 2.3727$ [12]. Both of these bounds take more than quadratic time, and more prohibitively quadratic space, making them unsuitable for modern data where the number of non-zeros in $\mathbf{A}$, $\mathrm{nnz}(\mathbf{A})$ is often 10^9 or more. As a result, there has been significant interest in either first-order methods with low per-step cost [13], [14], or faster algorithms taking advantage of additional structures of $\mathbf{A}$.

One case where significant runtime improvements are possible is when $\mathbf{A}$ is tall and thin, a.k.a. $n \gg d$. They appear in applications involving many data points in a smaller number of dimensions, or a few objects on which much data have been collected. These instances are sufficiently common that experimental speedups for finding QR factorizations of such matrices have been studied in the distributed [15], [16] and MapReduce settings [17]. The evidence for faster algorithms is perhaps more clear in the ℓ_2 setting, where finding $\mathbf{x}$ can be reduced to a linear system solve involving the $d \times d$ matrix $\mathbf{A}^T\mathbf{A}$. When $n \gg d$, the cost of inverting this matrix, $O(d^\omega)$ is less than the cost of examining the non-zeros in $\mathbf{A}$.

Faster algorithms for approximating $\mathbf{A}^T\mathbf{A}$ were first studied in the setting of approximation matrix multiplication [18], [19], [20]. Subsequent approaches were based on finding a shorter matrix $\mathbf{B}$ such that solving a regression problem on $\mathbf{B}$ leads to a similar answer [21], [1]. The running time of these routines were also gradually reduced [2], [4], [5], leading to algorithms that run in input sparsity time[6], [7]. These algorithms run in time proportional to the number of non-zeros in $\mathbf{A}$, $\mathrm{nnz}(\mathbf{A})$, plus a $\mathrm{poly}(d)$ term.

An approach common to these algorithms is that they reduce $\mathbf{A}$ to a $\mathrm{poly}(d)$ sized approximation using a single transformation. After that, any further steps only incur a $\mathrm{poly}(d)$ overhead. This is done by either obtaining high quality sampling probabilities [4], or by directly creating $\mathbf{B}$ via a sketching matrix [6], [7], [8]. These algorithms are appealing due to simplicity, speed, and that they can be adapted naturally in the streaming setting. On the other hand, experimental works have shown that practical performances are often optimized by applying higher error variants of these algorithms in an iterative fashion [22].

In this paper, we design algorithms motivated by these practical adaptations whose performances match or improve

	ℓ_2 Runtime for $\approx d$ Rows	ℓ_1	
		Runtime	# Rows
Dasgupta et al. [1]	-	$nd^5 \log d$	$d^{2.5}$
Magdon-Ismail [2]	$nd^2/\log d$	-	
Sohler and Woodruff [3]	-	$nd^{\omega-1+\theta}$	$d^{3.5}$
Drineas et al. [4]	$nd \log d$	-	
Clarkson et al. [5]	-	$nd \log d$	$d^{4.5} \log^{1.5} d$
Clarkson and Woodruff [6]	$\text{nnz}(\mathbf{A}) + d^3 \log d$	$\text{nnz}(\mathbf{A}) + d^7$	$d^8 \text{poly}(\log d)$
Mahoney and Meng [7]	$\text{nnz}(\mathbf{A}) + d^3 \log d$	$\text{nnz}(\mathbf{A}) \log n + d^8$	$d^{3.5}$
Nelson and Nguyen [8]	$\text{nnz}(\mathbf{A}) + d^{2+\theta}$	Similar to [6] and [7]	
This paper	$\text{nnz}(\mathbf{A}) + d^{\omega+\theta}$	$\text{nnz}(\mathbf{A}) + d^{\omega+\theta}$	$d^{3.66}$

Table I
COMPARISON OF RUNTIME AND SIZE OF $\mathbf{B}$ FOR ℓ_2 AND ℓ_1, θ IS ANY CONSTANT THAT'S > 0.

over the current best. Our algorithms construct $\mathbf{B}$ containing $\text{poly}(d)$ rows of $\mathbf{A}$ and run in $O(\text{nnz}(\mathbf{A}) + d^{\omega+\theta})$ time. Here the last term is due to computing inverses and change of basis matrices, and can be viewed as a lower order term since regression routines involving $d \times d$ matrices often take at least Ωd^ω time. In Table I we give a quick comparison of our results with previous ones in the ℓ_2 and ℓ_1 settings. These two norms encompass most of the regression problems solved in practice [10]. To simplify the comparison, we do not distinguish between $\log d$ and $\log n$, and assume that $\mathbf{A}$ has full column rank. We will also omit the big-O notation along with factors of ϵ and θ. In the ℓ_2 setting, many other algorithms [6], [7], [8] obtain an approximation using a sketching matrix. The number of rows of this matrix can be further reduced using other algorithms such as sampled fast Hadamard transform [4]. Therefore, the runtime needed to get to nearly d rows is perhaps more suitable for comparison. Here our results improve upon previous algorithms that run in input-sparsity time [6], [7], but the concurrent result by Nelson and Nguyen [8] gives a better bound. In the ℓ_1 setting, our algorithm gives clearly better additive terms.

As with previous results, our approaches and bounds for ℓ_2 and ℓ_p are fairly different. We will state them in more details and give a more detailed comparison with previous results in Section II. The key idea that drives our algorithms is that a constant factor reduction of problem size suffices for a linear time algorithm. This is a much weaker requirement than reducing directly to $\text{poly}(d)$ sized instances, and allows us to reexamine statistical projections with weaker guarantees. In the ℓ_2 setting, projections that do not even preserve the column space of $\mathbf{A}$ can still give good enough sampling probabilities. For the ℓ_p setting, estimating the probabilities in the 'wrong' norm (e.g. ℓ_2) still leads to significant reductions. Most of the subroutines that we'll use have either been used as the final error correction step [1], [5], [6], or are known in folklore. However, by combining these tools with techniques originally developed in graph sparsification and combinatorial preconditioning [23], we are able to convert them into much more powerful algorithms. A consequence of the simplicity of the routines used is that we obtain a smaller number of rows in $\mathbf{B}$ in the ℓ_2 setting, as well as a smaller running time in the ℓ_p setting.

Due to space constraints, we omit many details and proofs after the middle of Section IV-A. They can be found in the arXiv version of this paper [24], which is structured similarly.

II. OVERVIEW

We start by formalizing the requirements needed for $\mathbf{B}$ to be a good approximation to $\mathbf{A}$. In the ℓ_2 setting it is similar to $\mathbf{B}^T\mathbf{B}$ being an approximation to $\mathbf{A}^T\mathbf{A}$, but looking for $\mathbf{B}$ instead of $\mathbf{B}^T\mathbf{B}$ has the advantage of being extendible to ℓ_p norms [1]. The requirement for $\mathbf{B}$ is:

$$(1-\epsilon)\|\mathbf{Ax}\|_p \leq \|\mathbf{Bx}\|_p \leq (1+\epsilon)\|\mathbf{Ax}\|_p, \qquad \forall \mathbf{x} \in \mathbb{R}^d$$

Finding such a $\mathbf{B}$ is equivalent to reducing the size of a regression problem involving $\mathbf{A}$ since $\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_p = \min_{\mathbf{x}} \left\|[\mathbf{A}, \mathbf{b}][\mathbf{x}^T, -1]^T\right\|_p$ This means finding a shorter $(1 \pm \epsilon)$ approximation to the $n \times (d+1)$ matrix $[\mathbf{A}, \mathbf{b}]$, and solving a regression problem on this approximation gives a solution within $1 + O(\epsilon)$ of the minimum.

Row sampling is one of the first studied approaches for finding such $\mathbf{B}$ [21], [2], [1]. It aims to build $\mathbf{B}$ consisting of a set of rescaled rows of $\mathbf{A}$ chosen according to some distribution. While it appears to be an even more restrictive way of generating $\mathbf{B}$, it nevertheless leads to a row count within a factor of $\log d$ of the best known bounds [25], [26]. In ℓ_2, there exists a distribution that produces with high probability a good approximation $\mathbf{B}$ with $O(d \log d)$ rows [27], [28], [29], [30]; while under ℓ_p norm, $\text{poly}(d)$ rows is also known [1]. For a variety of objectives, these

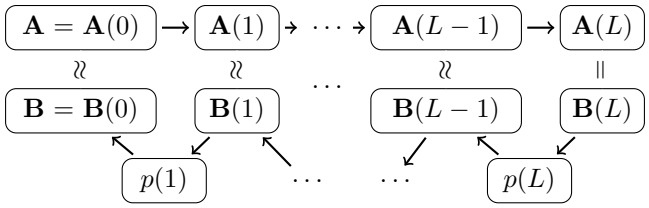

Figure 1. Main workflow of our algorithms when viewed as an iterative process. Sequence of gradually smaller matrices generated are on top, and the computed sampling probabilities and resulting approximations are below.

smaller equivalents have been studied as coresets [31], [32]. However, various properties of the ℓ_p norm, especially in the case of $p = 2$, makes row sampling a more specialized instance.

The main framework of our algorithm is iterative in nature and relies on the two-way connection between row sampling and estimation of sampling probabilities. A crude approximation $\mathbf{A}'$ to $\mathbf{A}$ allows us to compute equally crude approximations of sampling probabilities, while such probabilities in turn lead to higher quality approximations. The computation of these sampling probabilities can in turn be sped up using a high quality approximation of $\mathbf{A}'$. Our algorithm is based on observing that as long as $\mathbf{A}'$ has smaller size, we have made enough progress for an iterative algorithm. A single step in this algorithm consists of computing a small but crude approximation $\mathbf{A}'$, finding a higher quality approximation to $\mathbf{A}'$, and using this approximation to find estimates of sampling probabilities of the rows of $\mathbf{A}$. This leads to a tail-recursive process that can also be viewed as an iterative one where the calls generate a sequence of gradually shrinking matrices, and sampling probabilities are propagated back up the sequence. An example of such a sequence is given in Figure 1.

We will term the creation of the coarse approximation as **reduction**, and the computation of the more accurate approximation based on it as **recovery**. As in the figure, we will label the matrices $\mathbf{A}$ that we generate, as well as their approximations using the indices (l).

- reduction: creates a smaller version of $\mathbf{A}(l)$, $\mathbf{A}(l+1)$ with fewer rows either by a projection or a coarser row sampling process. It is equivalent to moving rightwards in Figure 1.
- recovery: finds a small, high quality approximation of $\mathbf{A}(l)$, $\mathbf{B}(l)$ using information obtained from $\mathbf{A}(l)$, $\mathbf{A}(l+1)$, and $\mathbf{B}(l+1)$. This is done by estimating sampling probabilities for the rows of $\mathbf{A}\,(l)$ using $\mathbf{B}\,(l+1)$, and sampling $\mathbf{A}\,(l)$ using these values. It is equivalent to moving leftwards in Figure 1.

Both our ℓ_2 and ℓ_p algorithms can be viewed as giving reduction and recovery routines. In the ℓ_2 setting our reduction step consists of a simple random projection, which incurs a fairly large distortion and may not even preserve the null space. Our key technical components in Section IV show that one-sided bounds on these projections are sufficient for recovery. This allows us to set the difference incurred by the reduction to $\kappa = d^{\theta}$ for an arbitrarily small $\theta > 0$, while obtaining a reduction factor of $\kappa^{O(1)} = d^{O(\theta)}$. This error is absorbed by the sampling process, and does not accumulate across the iterations.

Theorem 2.1: Given a $n \times d$ matrix $\mathbf{A}$ along with failure probability $\delta = d^{-c}$ and allowed error ϵ. For any constant $\theta > 0$, we can find in $O(\mathrm{nnz}(A)+d^{\omega+\theta}\epsilon^{-2})$ time, with probability at least $1-\delta$, a matrix $\mathbf{B}$ consisting of $O(d \log d\epsilon^{-2})$ rescaled rows of $\mathbf{A}$ such that

$$(1-\epsilon)\|\mathbf{Ax}\|_2 \leq \|\mathbf{Bx}\|_2 \leq (1+\epsilon)\|\mathbf{Ax}\|_2$$

for all vectors $\mathbf{x} \in \mathbb{R}^d$.

This bound improves the $O(d^2)$ rows obtained in the first results with input-sparsity runtime [6], and matches the best bound known using oblivious projections [8], which was obtained concurrently. A closer comparison with [8] shows that our bounds does not have a factor of ϵ^{-1} on the leading term $\mathrm{nnz}(\mathbf{A})$, but has worse dependencies on θ.

For ℓ_p norms, we show that significant size reductions can be made if we perform row sampling using sampling probabilities obtained in a different norm. Specifically, if $\mathbf{A}(i)$ has $n(i)$ rows, $\mathbf{A}(i+1)$ has $O(n(i)^{c_p}\mathrm{poly}(d))$ where $c_p < 1$ if the intermediate norm $\ell_{p'}$ is chosen appropriately. This means the number of rows will reduce doubly exponentially as we iterative, and quickly becomes $O(\mathrm{poly}(d))$.

This allows us to compute these probabilities using the algorithms for ℓ_2 from Section IV, as well as $\ell_{p'}$ approximations under different norms. Since these routines already produce samples with $\mathrm{poly}(d)$ rows, the algorithm is also more direct. Instead of gradually going back up the sequence of matrices, we recover an approximation to $\mathbf{A}$ after each iteration

Our projection and recovery methods are similar to the ones used for increasing the accuracy of ℓ_1 row sampling in [5]. However, our result is the first that uses such steps in an iterative manner, and also the first to use them as the the core component. They show a much tighter connection between ℓ_2 and ℓ_p row sampling, namely that finding good ℓ_2 approximations alone is sufficient for iterative reductions in matrix size. In Section V-A we present an one step variant that computes sampling probabilities under the ℓ_2 norm. It gives $\mathbf{B}$ with about $d^{\frac{4}{p}}$ rows when $p \leq 2$ and $d^{\frac{3p-2}{4-p}}$ rows when $2 \leq d < 4$. We can further iterate upon this algorithm, and compute sampling probabilities under the $\ell_{p'}$ norm for some p' between 2 and p. A two-level version of this algorithm for ℓ_1 is analyzed in Section V-B, giving the following:

Theorem 2.2: Given a $n \times d$ matrix $\mathbf{A}$ along with failure probability d^{-c} and allowed error ϵ. For any constant $\theta > 0$, we can find in $O(\mathrm{nnz}(A)+d^{\omega+\theta}\epsilon^{-2})$ time, with probability

at least $1 - d^{-c}$, a matrix $\mathbf{B}$ consisting of $O(d^{4\sqrt{2}-2+\theta})$ rescaled rows of $\mathbf{A}$ such that

$$(1-\epsilon)\left\|\mathbf{A}\mathbf{x}\right\|_1 \leq \left\|\mathbf{B}\mathbf{x}\right\|_1 \leq (1+\epsilon)\left\|\mathbf{A}\mathbf{x}\right\|_1$$

for all vectors $\mathbf{x} \in \mathbb{R}^d$.
This method readily leads to $\mathbf{B}$ with $\mathrm{poly}(d)$ rows when $p \geq 4$, and fewer rows than the above bound when $1 \leq p < 4$. However, such extensions are limited by the discontinuity between bounds on the sampling process in the ℓ_2 [27], [28], [29], [30] and ℓ_p settings [1]. As a result, we only show the algorithm for ℓ_1 in order to simplify the presentation.

An additional benefit of our approach is that the randomized routines used hold with high probability. Most of the earlier results that run in time nearly-linear in the size of $\mathbf{A}$ have a constant success probability, and increasing it often leads to a $\log d$ overhead on the leading term. Also, as our algorithm is row sampling based, each row in our output is a scaled copy of some row of the original matrix. This means specialized structure for rows of $\mathbf{A}$ are likely to be preserved in the smaller regression problem instance.

The main drawback of our algorithm in the ℓ_2 setting is that it does not immediately extend to computing low-rank approximations. The method given in [6] relies crucially on first transform being oblivious, although our algorithm can be incorporated in a limited way as the second step. Also, the algorithm is non-streaming, and more complicated than the sketching based approaches for input-sparsity time algorithms given in [6], [7], [8]. For ℓ_p row-sampling, our algorithms invoke concentration bounds from [1] in a black-box manner, even though our sampling probabilities are obtained in the ℓ_2 setting. We believe investigating whether our algorithm can be combined with other approaches, extending our approaches to low-rank approximations, and obtaining tighter concentration bounds are natural directions for future work.

III. Preliminaries

We begin by stating key notations and definitions that we will use for the rest of this paper. Our algorithms and analysis use standard linear algebraic notations as given in [9]. We will use $\|\mathbf{x}\|_p$ to denote the ℓ_p norm of a vector. Two values of p that we'll use frequently are $p = 1$ and $p = 2$, which correspond to $\|\mathbf{x}\|_1 = \sum_i |x_i|$ and $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$. For two vectors $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x} \geq \mathbf{y}$ means $\mathbf{x}$ is entry-wise greater or equal to $\mathbf{y}$, a.k.a. $\mathbf{x}_i \geq \mathbf{y}_i$ for all i.

For a matrix $\mathbf{A}$, we use $\mathbf{A}_{i*}$, or $\mathbf{a}_i$ to denote the i^{th} row of $\mathbf{A}$, and $\mathbf{A}_{*j}$ to denote its j^{th} column. Note that if $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\mathbf{a}_i$ is a row vector of length d. We will also use the generalized ℓ_p-norm $\left|\left|\left|\cdot\right|\right|\right|_p$ of a matrix, which essentially treats all entries of the matrix as a single vector. Specifically, $\left|\left|\left|\mathbf{A}\right|\right|\right|_p = (\sum_{ij} |\mathbf{A}_{ij}|^p)^{1/p}$. When $p = 2$, it is known as the Frobenius norm, $\|\cdot\|_F$.

Matrices such as $\mathbf{A}^T\mathbf{A}$ are symmetric, and have identical row/column spaces, and in turn identical left/right null spaces. Such a matrix, $\mathbf{C}$, is positive semi-definite if all its eigenvalues are non-negative, or equivalently $\mathbf{x}^T\mathbf{C}\mathbf{x} \geq 0$ for all vectors $\mathbf{x}$. Similarity between symmetric matrices can be defined using a partial order that generalizes $\leq$. For two matrices $\mathbf{C}_1$ and $\mathbf{C}_2$, $\mathbf{C}_1 \preceq \mathbf{C}_2$ denotes that $\mathbf{C}_2 - \mathbf{C}_1$ is positive semi-definite. The connection between this notation and row sampling is clear in the case of ℓ_2, specifically $\mathbf{A}^T\mathbf{A} \preceq \mathbf{B}^T\mathbf{B}$ is equivalent to $\|\mathbf{A}\mathbf{x}\|_2 \leq \|\mathbf{B}\mathbf{x}\|_2$.

We will also define the pseudoinverse of $\mathbf{C}$, $\mathbf{C}^\dagger$ as the linear operator that's zero on the null space of $\mathbf{C}$, while acting as its inverse on the column space. For operators that act on the same space, spectral orderings of pseudoinverses can be inverted in the same way as scalars. Specifically, if $\mathbf{C}_1$ and $\mathbf{C}_2$ have the same null space and $\mathbf{C}_1 \preceq \mathbf{C}_2$, then $\mathbf{C}_2^\dagger \preceq \mathbf{C}_1^\dagger$. Given a subspace of $\mathbb{R}^d$, an orthogonal projector onto it, $\mathbf{\Pi}$ is a symmetric positive-semidefinite matrix taking vectors into their projection in this space. For example, if this space is the column space of some positive semi-definite matrix $\mathbf{C}$, then an orthogonal projection operator is given by $\mathbf{C}\mathbf{C}^\dagger$.

Our algorithms are designed around the following algorithmic fact: for any norm p and any matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, there exists a distribution on its rows such that sampling $\mathrm{poly}(d)$ entries from this distribution and rescaling them gives $\mathbf{B}$ such that with probability at least $1 - d^{-c}$:

$$(1-\epsilon)\|\mathbf{A}\mathbf{x}\|_p \leq \|\mathbf{B}\mathbf{x}\|_p \leq (1+\epsilon)\|\mathbf{A}\mathbf{x}\|_p, \qquad \forall \mathbf{x} \in \mathbb{R}^d.$$

This sampling process can be formalized in several ways, leading to similar results both theoretically and experimentally [33]. We will treat it as a black-box $\textsc{Sample}(\mathbf{A}, \mathbf{p})$ that takes a set of probabilities over the rows of $\mathbf{A}$ and samples them accordingly. It keeps row i or $\mathbf{A}$ with probability $\min\{1, p_i\}$, and rescales it appropriately so the expected value of this row is preserved. The two key properties of $\textsc{Sample}(\mathbf{A}, \mathbf{p})$ that we will use repeatedly are:

- It returns $\mathbf{B}$ with at most $O(|\mathbf{p}|_1)$ rows.
- Its running time can be bounded by $O(n + |\mathbf{p}|_1 \log n)$.

The convergence of sampling relies on matrix Chernoff bounds, which can be viewed as generalizations of single variate concentration bounds [34]. Necessary conditions on the probabilities can be formalized in several ways, with the most common being statistical leverage scores. These values have been well studied in statistics, and their use in algorithms has been more recent [35]. The most general definition of ℓ_p-norm leverage scores is based on the row norms of a basis of the column space of $\mathbf{A}$. However, significant simplifications are possible when $p = 2$, and this alternate view is crucial in our algorithm. As a result, we will state the relevant convergence results for $\textsc{Sample}$ separately in Sections IV and V.

These concentration bounds show that statistical leverage scores are closely associated the probabilities needed for row sampling, and give algorithms that efficiently approximate

these values. We will also formalize an observation implicit in previous results that both the sampling and estimation algorithms are highly robust. The high error-tolerance of these algorithms makes them ideal as core routines to build iterative algorithms upon.

One issue with the various concentration bounds that we will prove is that they hold with high probability in d. In other words, they fail with probability $1 - d^{-c}$ for some constant c. In cases where $n \gg \text{poly}(d)$, this will prevent us from taking an union bound over many sampling steps. However, it can be shown that in such cases, increasing all sampling probabilities with $1/\text{poly}(d)$ in the sampling process narrows the number of rows of interest back to $\text{poly}(d)$. This leads to an approximation with $n' = O(\text{nnz}(\mathbf{A})/\text{poly}(d))$ rows. This reduction in row count allows us to then apply routines that runs in $O(n'\text{poly}(d)) = O(\text{nnz}(\mathbf{A}))$ time (e.g. [1]). Such routines then lead to approximations with $\text{poly}(d)$ rows with high probability. Therefore for the rest of this paper we will make the simplifying assumption that $n = \text{poly}(d)$.

IV. Iterative Row Sampling for Preserving ℓ_2-Norm

We start by presenting our algorithm for computing row sampling in the ℓ_2 setting. Crucial to our approach is the following basis-free definition of statistical leverage scores of $\boldsymbol{\tau}$:

$$\tau_i \stackrel{\text{def}}{=} \mathbf{a}_i(\mathbf{A}^T\mathbf{A})^\dagger\mathbf{a}_i^T, \quad \text{for } i = 1, \ldots, n,$$

where $\mathbf{a}_i$ is the i-th row of $\mathbf{A}$.

To our knowledge, the first nearly tight bounds for row sampling using statistical leverage scores were given in [27], and various extensions and simplifications were made since [28], [29], [30], [36]. The state-of-the-art bound as given in [34] can be stated as:

Lemma 4.1: If $\tilde{\boldsymbol{\tau}}$ is a set of probabilities such that $\tilde{\boldsymbol{\tau}} \geq \boldsymbol{\tau}$, then for any constants c and ϵ, there exists a function $\text{SAMPLE}(\mathbf{A}, O(\log d, \epsilon)\tilde{\boldsymbol{\tau}})$ which returns $\mathbf{B}$ containing $O(\log d \left\|\tilde{\boldsymbol{\tau}}\right\|_1 \epsilon^{-2})$ rows and satisfying

$$(1-\epsilon)\|\mathbf{A}\mathbf{x}\|_2 \leq \|\mathbf{B}\mathbf{x}\|_2 \leq (1+\epsilon)\|\mathbf{A}\mathbf{x}\|_2, \qquad \forall \mathbf{x} \in \mathbb{R}^d$$

with probability at least $1 - d^{-c}$.

The importance of statistical leverage scores can be reflected in the following fact, which implies that we can obtain $\mathbf{B}$ with $O(d \log d)$ rows.

Fact 4.2: Given $n \times d$ matrix $\mathbf{A}$, and let $\boldsymbol{\tau}$ be its leverage scores w.r.t. $\mathbf{A}$. Assume $\mathbf{A}$ has rank r, then

$$\sum_{i=1}^{n} \tau_i = r \leq d$$

Although it is tempting to directly obtain high quality approximations of the leverage scores, their computation also requires a high quality approximation of $\mathbf{A}^T\mathbf{A}$, leading us back to the original problem of row sampling. Our way around this issue relies on the robustness of concentration bounds such as Lemma 4.1. Sampling using even crude estimates on leverage scores can lead to high quality approximations [1], [37], [38], [4], [36]. Therefore, we will not approximate $\mathbf{A}^T\mathbf{A}$ directly, and instead obtain a sequence of gradually better approximations. The need to compute sampling probabilities using crude approximations leads us to define a generalization of statistical leverage scores.

A. Generalized Stretch and its Estimation

Using different matrices to upper bound leverage scores is implicit in many previous algorithms [37], [5]. One area where it plays a crucial role is combinatorial preconditioning, where such upper bounds measured w.r.t. a tree is termed stretch [39], [23]. We will draw from them and term our generalization of leverage scores **generalized stretch**. We will use $\text{STR}_{\mathbf{B}}(\mathbf{a}_i)$ to denote the approximate leverage score of row i computed as follows:

$$\text{STR}_{\mathbf{B}}(\mathbf{a}_i) \stackrel{\text{def}}{=} \mathbf{a}_i(\mathbf{B}^T\mathbf{B})^\dagger\mathbf{a}_i^T \tag{4.1}$$

Under this definition, the original definition of statistical leverage score τ_i equals to $\text{STR}_{\mathbf{A}}(\mathbf{a}_i)$. We will refer to $\mathbf{B}$ as the reference used to compute stretch. It can be shown that when $\mathbf{B}_1$ and $\mathbf{B}_2$ are reasonably close to each other, stretch can be used as upper bounds for leverage scores in a way that satisfies Lemma 4.1.

Lemma 4.3: If $\mathbf{B}_1$ and $\mathbf{B}_2$ satisfies:

$$\frac{1}{\kappa}\mathbf{B}_1^T\mathbf{B}_1 \preceq \mathbf{B}_2^T\mathbf{B}_2 \preceq \mathbf{B}_1^T\mathbf{B}_1$$

Then for any vector $\mathbf{x}$ we have:

$$\text{STR}_{\mathbf{B}_1}(\mathbf{x}) \leq \text{STR}_{\mathbf{B}_2}(\mathbf{x}) \leq \kappa\text{STR}_{\mathbf{B}_1}(\mathbf{x})$$

The stretch notation also extends to a set of rows, or an entire matrix. If $\mathbf{A}$ is a matrix with n rows, $\text{STR}_{\mathbf{B}}(\mathbf{A})$ denotes:

$$\text{STR}_{\mathbf{B}}(\mathbf{A}) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \text{STR}_{\mathbf{B}}(\mathbf{a}_i). \tag{4.2}$$

This multiplication by $(\mathbf{B}\mathbf{B}^T)^{\dagger\frac{1}{2}}$ can be viewed as transforming the space into an isotropic position. This transformation decorrelates the rows of $\mathbf{B}$, and makes the stretch of each row the ℓ_2^2 norm of a corresponding vector. The stretch of a set of rows also corresponds to the square of the Frobenius norm of their corresponding matrix.

Fact 4.4: The generalized stretch of the i-th row of $\mathbf{A}$ w.r.t $\mathbf{B}$ equals to its ℓ_2^2 norm under the transformation $(\mathbf{B}\mathbf{B}^T)^{\dagger\frac{1}{2}}$:

$$\text{STR}_{\mathbf{B}}(\mathbf{a}_i) = \|(\mathbf{B}\mathbf{B}^T)^{\dagger\frac{1}{2}}\mathbf{a}_i^T\|_2^2 = \|\mathbf{a}_i(\mathbf{B}\mathbf{B}^T)^{\dagger\frac{1}{2}}\|_2^2$$

and the total stretch of all rows is

$$\text{STR}_{\mathbf{B}}(\mathbf{A}) = \|(\mathbf{B}\mathbf{B}^T)^{\dagger\frac{1}{2}}\mathbf{A}^T\|_F^2 = \|\mathbf{A}(\mathbf{B}\mathbf{B}^T)^{\dagger\frac{1}{2}}\|_F^2$$

This representation leads to faster algorithms for estimating stretch using the Johnson-Lindenstrauss transform. This tool is used in a variety of settings from estimating effective resistances [40] to more generally leverage scores [4]. We will use the following randomized projection theorem:

Lemma 4.5: (Lemma 2.2 from [41]) Let $\mathbf{y}$ be an unit vector in $\mathbb{R}^d$. Then for any positive integer $k \leq d$, let $\mathbf{U}$ be a $k \times d$ matrix with entries chosen independently from the Gaussian distribution $\mathcal{N}(0,1)$. Let $\mathbf{x} = \mathbf{U}\mathbf{y}$ and $L = \|\mathbf{x}\|_2^2$. Then for any $R > 1$,

1) $\mathbb{E}(L) = k$
2) $\Pr\left(L \geq Rk\right) < \exp\left(\frac{k}{2}(1 - R + \ln R)\right)$
3) $\Pr\left(L \leq \frac{k}{R}\right) < \exp\left(\frac{k}{2}(1 - R^{-1} - \ln R)\right)$

We will also use this lemma in our reduction step to bound the distortion when rows are combined. Note that the requirements of Lemma 4.1 and the guarantees of SAMPLE allows our estimates to have larger error. This means we can use fewer vectors in the projection, and scale up the results to correct potential underestimates. Therefore, we can trade the coefficient on the leading term $\mathrm{nnz}(\mathbf{A})$ with a higher number of sampled row count. To simplify the algorithm, we encapsulate this use of the the Johnson-Lindenstrauss transform and the incorporation of $\mathbf{B}$ into a single routine.

Lemma 4.6: For any constant c, there is a routine $\textsc{ApxStr}(\mathbf{A}, \mathbf{B}, \kappa, R)$ that when given a $n \times d$ matrix $\mathbf{A}$ where $n = \mathrm{poly}(d)$, and an approximation $\mathbf{B}$ with m rows such that:

$$\frac{1}{\kappa}\mathbf{A}^T\mathbf{A} \preceq \mathbf{B}^T\mathbf{B} \preceq \mathbf{A}^T\mathbf{A}$$

APXSTR returns in $O((\mathrm{nnz}(A)+d^2)\log_R d + (m+d)d^{\omega-1})$ time and upper bounds $\tilde{\tau}_i$ such that with probability at least $1 - d^{-c}$

1) for all i, $\tilde{\tau}_i \geq \tau_i$.
2) $\|\tilde{\boldsymbol{\tau}}\|_1 \leq O(R^2\kappa d)$.

B. Reductions and Recovery

Our reduction and recovery processes are based on projecting $\mathbf{A}$ to a matrix with fewer rows, and moving the estimates on the projection back to the original matrix. Our key operation is to combine every R rows into k rows, where R and k are set to d^θ and $O(c/\theta)$ respectively. By padding $\mathbf{A}$ with additional rows of zeros, we may assume that the number of rows is divisible by R. We will use $n_b = n/R$ to denote the number of blocks, and use the notation $\cdot_{(b)}$ to index into the b^{th} block. Our key step is then a (R,k)-reduction of the rows:

Definition 4.7: A (R,k)-reduction of $\mathbf{A}$ describes the following procedure:

1) For each block $\mathbf{A}_{(b)}$, pick $\mathbf{U}_{(b)}$ to be a $k \times R$ random Gaussian matrix with entries picked independently from $\mathcal{N}(0,1)$ and compute $\mathbf{A}_{\downarrow(b)} = \mathbf{U}_{(b)}\mathbf{A}_{(b)}$.
2) Concatenate the blocks $\mathbf{A}_{\downarrow(b)}$ together vertically to form $\mathbf{A}_\downarrow$.

We first show that projections preserve the stretch of blocks w.r.t. $\mathbf{A}$. This can be done by bounding the effect of $\mathbf{U}_{(b)}$ on the norm of each column of $\mathbf{A}_{(b)}(\mathbf{A}^T\mathbf{A})^{\dagger\frac{1}{2}}$. It follows directly from properties of the Johnson-Lindenstrauss projections described in Lemma 4.5:

Lemma 4.8: Assume $R = d^\theta \geq e^2$ for some constant θ and let $\mathbf{A}_\downarrow$ be a (R,k)-projection of $\mathbf{A}$. For any constant $c > 0$ there exists a constant $k = O(c/\theta)$ such that:

$$\mathrm{STR}_{\mathbf{A}}(\mathbf{A}_{\downarrow(b)}) \geq \frac{k}{R}\mathrm{STR}_{\mathbf{A}}(\mathbf{A}_{(b)})$$

holds for all blocks $b = 1, \ldots, n_b$ with probability at least $1 - d^{-c}$.

We next show that we can change the reference from $\mathbf{A}$ to $\mathbf{A}_\downarrow$, and use $\mathrm{STR}_{\mathbf{A}_\downarrow}(\mathbf{A}_{\downarrow(b)})$ as upper bounds for $\mathrm{STR}_{\mathbf{A}}(\mathbf{A}_{\downarrow(b)})$. As a first step, we need to relate $\mathbf{A}^T\mathbf{A}$ to $\mathbf{A}_\downarrow{}^T\mathbf{A}_\downarrow$. Since each $\mathbf{A}_{\downarrow(b)}$ is formed by merging rows of $\mathbf{A}_{\downarrow(b)} = \mathbf{U}_{(b)}\mathbf{A}_{(b)}$, $\mathbf{A}_{\downarrow(b)}^T\mathbf{A}_{\downarrow(b)}$ can be upper bounded by $\mathbf{A}_{(b)}^T\mathbf{A}_{(b)}$ times a suitable term depending on $\mathbf{U}_{(b)}$.

Lemma 4.9: The following holds for each block b:

$$\mathbf{A}_{\downarrow(b)}^T\mathbf{A}_{\downarrow(b)} \preceq \|\mathbf{U}_{(b)}\|_F^2 \cdot \mathbf{A}_{(b)}^T\mathbf{A}_{(b)}$$

However, generalized stretches w.r.t. $\mathbf{A}$ and $\mathbf{A}_\downarrow$ are evaluated under the norms given by the pseudoinverses of these matrices, $(\mathbf{A}^T\mathbf{A})^\dagger$ and $(\mathbf{A}_\downarrow{}^T\mathbf{A}_\downarrow)^\dagger$. As a result, we need to bound the difference between these two pseudoinverses spectrally using the following lemma.

Lemma 4.10: Let $\mathbf{C}$ and $\mathbf{D}$ be symmetric positive semidefinite matrices and let $\mathbf{\Pi}$ be the orthogonal projection operator onto the column space of $\mathbf{C}$. Then:

$$\mathbf{\Pi}\mathbf{C}^\dagger\mathbf{\Pi} \succeq \mathbf{\Pi}(\mathbf{C} + \mathbf{D})^\dagger\mathbf{\Pi}$$

This is straightforward when both $\mathbf{C}$ and $\mathbf{D}$ are full rank, or share the same null space. However, as pseudoinverses do not act on the null space, it is crucial that we're only considering vectors of the form $\mathbf{a}_i'$. Combining it with bounds in the other direction allows us to bound the distortion caused by switching the reference from $\mathbf{A}$ to $\mathbf{A}_\downarrow$.

Lemma 4.11: For any constant c, there exists a constant c', such that with probability at most $1 - d^{-c}$, we have for each row i of $\mathbf{A}_\downarrow$, denoted by $\mathbf{a}_{\downarrow i}$, satisfies

$$c'kR\log d \cdot \mathrm{STR}_{\mathbf{A}_\downarrow}(\mathbf{a}_{\downarrow i}) \geq \mathrm{STR}_{\mathbf{A}}(\mathbf{a}_{\downarrow i})$$

Combining Lemmas 4.11 and 4.8 shows that with high probability, scaling up $\mathrm{STR}_{\mathbf{A}_\downarrow}(\mathbf{A}_{\downarrow(b)})$ by $O(R^2\log d)$ gives upper bounds for the leverages scores in the original blocks of $\mathbf{A}$.

Corollary 4.12: For any constant c, there exists a setting of constants such that for any $R = d^\theta$, we have with probability at least $1 - d^{-c}$

$$c'R^2\log d \cdot \mathrm{STR}_{\mathbf{A}_\downarrow}(\mathbf{A}_{\downarrow(b)}) \geq \mathrm{STR}_{\mathbf{A}}(\mathbf{A}_{(b)})$$

holds for all b.

C. Iterative Algorithm

It remains to make use of the estimates that we obtain using this projection process. Projecting $\mathbf{A}$ to $\mathbf{A}_{\downarrow}$ gives a matrix with fewer rows, and a way to reduce the sizes of our problems. A fast algorithm follows by examining the sequence of matrices $\mathbf{A} = \mathbf{A}(0), \mathbf{A}(1), \ldots \mathbf{A}(L)$ obtained using such projections. Once $\mathbf{A}(L)$ has fewer than $\mathrm{nnz}(\mathbf{A})d^{-3}$ rows, $\mathbf{A}(L)^T\mathbf{A}(L)$ can be approximated directly. This then allows us to approximate the statistical leverage scores of the rows of $\mathbf{A}(L)$ Corollary 4.12 shows that stretches computed on $\mathbf{A}(l)$, $\tilde{\tau}(l)$ can serve as sampling probabilities in $\mathbf{A}(l-1)$. This means we can gradually propagate solutions backwards from $\mathbf{A}(L)$ to $\mathbf{A}(0)$. We do so by maintaining the invariant that $\mathbf{B}(l)$ has a small number of rows and is close to $\mathbf{A}(l)$. The total generalized stretch of $\mathbf{A}(l)$ w.r.t. $\mathbf{B}(l)$ can be used as upper bounds of the statistical leverage scores of $\mathbf{A}(l-1)$ after suitable scaling.. This allows the sampling process to compute $\mathbf{B}(l-1)$, keeping the invariant for $l-1$. The algorithm is illustrated in Figure 2, and its pseudocode is given in Algorithm 1.

Algorithm 1 Row Sampling using Projections

RowSampleL2$(\mathbf{A}, R, \epsilon)$

Input: Reduction rate R, $n \times d$ matrix $\mathbf{A}$, allowed approximation error ϵ, failure probability $\delta = d^{-c}$.

Output: Sparsifier $\mathbf{B}$ that contains $O(R^5 d \log d/\epsilon^2)$ scaled rows of $\mathbf{A}$ such that $(1-\epsilon)\mathbf{A}^T\mathbf{A} \preceq \mathbf{B}^T\mathbf{B} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A}$.

Set $L = \lceil \log_R(n/d) \rceil$
Set $\epsilon(0) = \epsilon/3$, $\epsilon(1) \ldots \epsilon(l) = 1/2$
$\mathbf{A}(0) = \mathbf{A}$
for $l = 1 \ldots L$ **do**
  Let $\mathbf{A}(l)$ be a (R,k)-projection of $\mathbf{A}(l-1)$
end for
$\mathbf{B}(L) \leftarrow \mathbf{A}(L)$
for $i = L \ldots 1$ **do**
  $\tilde{\boldsymbol{\tau}}'(l) \leftarrow O(R^3 \log d) \cdot \text{ApxStr}(\mathbf{A}(l), \sqrt{\frac{2}{3}}\mathbf{B}(l), R, R)$
  Set each entry in $\tilde{\tau}(l-1)_{(b)}$ to $|\tilde{\boldsymbol{\tau}}'(l)_{(b)}|_1$
  $\mathbf{B}(l-1) \leftarrow \text{Sample}(\mathbf{A}(l-1), \tilde{\tau}(l-1), \epsilon(l))$
end for
$\tilde{\boldsymbol{\tau}}'(0) \leftarrow \text{ApxStr}(\mathbf{B}(0), \mathbf{B}(0), 2, 2)$
return $\text{Sample}(\mathbf{B}(0), \tilde{\boldsymbol{\tau}}'(0), \epsilon/3)$

Sampling probabilities for $\mathbf{A}(l-1)$ are obtained by computing the stretch of $\mathbf{A}(l)$ w.r.t. $\mathbf{B}(l)$. We first show that these values, $\tilde{\tau}(l-1)$, are with high probability upper bounds for the statistical leverage scores of $\mathbf{A}(l-1)$, $\tau(l-1)$.

Lemma 4.13: If $\mathbf{A}(l)$ and $\mathbf{B}(l)$ satisfy:

$$\frac{1}{2}\mathbf{A}(l)^T\mathbf{A}(l) \preceq \mathbf{B}(l)^T\mathbf{B}(l) \preceq \frac{3}{2}\mathbf{A}(l)^T\mathbf{A}(l)$$

then for any constant c, there is a setting of the constants in the algorithm such that:

- $\tilde{\boldsymbol{\tau}}(l-1) \geq \boldsymbol{\tau}(l-1)$

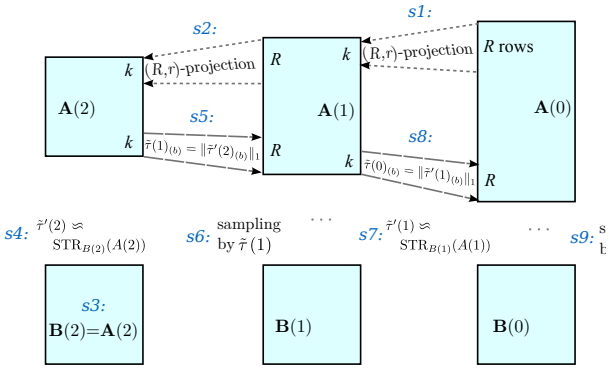

Figure 2. Illustration of Algorithm 1 by using $L = 2$. There are mainly two stages with 9 steps. On the reduction stage, we obtain shorter $\mathbf{A}(1)$ and $\mathbf{A}(2)$ by iteratively doing (R,r)-projection. On the next recovery stage, we approximate the leverage scores of $\mathbf{A}(1)$ by the ones computed from $\mathbf{A}(2)$ and $\mathbf{B}(2)$. Then $\mathbf{B}(1)$ is sampled based on this approximated scores, which will be used to further obtain the approximated leverage scores of $A(0)$. The final step which samples $\mathbf{B}(0)$ once again is not shown here.

- $\|\tilde{\boldsymbol{\tau}}(l-1)\|_1 \leq O(dR^3 \log d)$

holds with probability at least $1 - d^{-c}$.

Combining these with the fact that the number of rows decrease by a factor of $O(R)$ per iteration completes the algorithm. Our main result for ℓ_2 row sampling is obtained by setting R to d^θ. Applying Lemma 4.13 inductively backwards in l gives the overall bound.

Theorem 4.14: For any constant c, there is a setting of constants such that if RowSampleL2, shown in Algorithm 1, is run with $R = d^\theta$, then with probability at least $1 - d^{-c}$ it returns $\mathbf{B}$ in $O(\mathrm{nnz}(\mathbf{A}) + d^{\omega+4\theta}\epsilon^{-2})$ time such that:

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} \preceq \mathbf{B}^T\mathbf{B} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A}$$

and $\mathbf{B}$ has $O(d \log d \epsilon^{-2})$ rows, each being a scaled copy of some row of $\mathbf{A}$,

V. Algorithm for Preserving ℓ_p norm

We now turn to the more general problem of finding $\mathbf{B}$ with $\mathrm{poly}(d)$ rows such that:

$$(1-\epsilon)\|\mathbf{Ax}\|_p \leq \|\mathbf{Bx}\|_p \leq (1+\epsilon)\|\mathbf{Ax}\|_p \qquad \forall \mathbf{x} \in \mathbb{R}^d$$

We will make repeated use of the following (tight) inequality between ℓ_2 and ℓ_p norms, which can be obtained by direct applications of power-mean and Hölder's inequalities.

Fact 5.1: Let $\mathbf{x}$ be any vector in $\mathbb{R}^d$, and p and q any two norms where $1 \leq p \leq q$, we have:

$$\|\mathbf{x}\|_q \leq \|\mathbf{x}\|_p \leq d^{\frac{1}{q}-\frac{1}{p}}\|\mathbf{x}\|_q$$

We will use this Fact with one of p or q being 2, in which case it gives:

- If $1 \leq p \leq 2$, $\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_p \leq d^{\frac{1}{2}-\frac{1}{p}}\|\mathbf{x}\|_2$
- If $2 \leq p$, $d^{\frac{1}{p}-\frac{1}{2}}\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_p \leq \|\mathbf{x}\|_2$

As $\mathbf{Ax}$ is a length n vector, its ℓ_2 and ℓ_p norms may differ by a factor of $\mathrm{poly}(n)$. This means that the ℓ_2 row sampling algorithm from Section IV can lead to $\mathrm{poly}(n)$ distortion. Our algorithm in this section reduces this distortion iteratively. Once again, it is built around a concentration bound for a sampling process. The sampling probabilities are based on the definition of a well-conditioned basis, which is more flexible than ℓ_2 statistical leverage scores.

Definition 5.2: Let $\mathbf{A}$ be a $n \times d$ matrix of rank r, $p \in [1, \infty]$ and q be its dual norm such that $\frac{1}{p} + \frac{1}{q} = 1$. Then an $n \times r$ matrix $\mathbf{U}$ is an (α, β, p)-well-conditioned basis for the column space of $\mathbf{A}$ if the columns of $\mathbf{U}$ span the column space of $\mathbf{A}$ and:

1) $|||\mathbf{U}|||_p \leq \alpha$.
2) For all $\mathbf{x} \in \mathbb{R}^r$, $\|\mathbf{x}\|_q \leq \beta \|\mathbf{Ux}\|_p$.

A ℓ_p analog of the sampling concentration result given in Lemma 4.1 was shown in [1]. It can be viewed a generalization of Lemma 4.1.

Lemma 5.3: (Theorem 6 of [1]) Let $\mathbf{A}$ be a $n \times d$ matrix with rank r, $\epsilon \leq 1/7$, and let $p \in [1, \infty)$. Let $\mathbf{U}$ be an (α, β, p)-well-conditioned basis for $\mathbf{A}$. Then for any sampling probabilities $\mathbf{p} \in \mathbb{R}^n$ such that:

$$p_i \geq c_p (\alpha\beta)^p \frac{\|\mathbf{U}_{i*}\|_p^p}{|||\mathbf{U}|||_p^p},$$

where $\mathbf{U}_{i*}$ is the i-th row of $\mathbf{U}$ and c_p is a constant depending only on p. Then with probability at least $1 - d^{-c}$, SAMPLE$(\mathbf{A}, p, \epsilon)$ returns $\mathbf{B}$ satisfying

$$(1-\epsilon)\|\mathbf{Ax}\|_p \leq \|\mathbf{Bx}\|_p \leq (1+\epsilon)\|\mathbf{Ax}\|_p \qquad \forall \mathbf{x} \in \mathbb{R}^d$$

We omitted the reductions of probabilities that are more than 1 since this step is included in our formulation of SAMPLE. Several additional steps are needed to turn this into an algorithmic routine, the first being computing $\mathbf{U}$. A naïve approach for this requires matrix multiplication, and the size of the outcome may be more than $\mathrm{nnz}(\mathbf{A})$. Alternatively, we can find a linear transformation that leads to $\mathbf{U}$; specifically, a matrix $\mathbf{C}$ such that $\mathbf{U} = \mathbf{AC}$.

The estimation of $\|(\mathbf{AQ})_{i*}\|_p^p$ and sampling can then be done in a way similar to Section IV. When $1 \leq p \leq 2$, we can compute $O(1)$ approximations using p-stable distributions [42] in a way analogous to Section 4.2.1. of [5]. When $2 \leq p$, we will use the ℓ_2 norm as a surrogate at the cost of more rows. We encapsulate the estimation of ℓ_p norm leverage scores and sampling as a single black-box.

Lemma 5.4: For any constant c, there is an algorithm ESTIMATEANDSAMPLEP$(\mathbf{A}, \mathbf{C}, \alpha, \beta, R, \epsilon)$ that given matrices $\mathbf{A}$, $\mathbf{C}$ such that $\mathbf{AC}$ is a (α, β, p)-well-conditioned basis for $\mathbf{A}$, returns a matrix $\mathbf{B}$ with probability at least $1 - d^{-c}$ such that:

$$(1-\epsilon)\|\mathbf{Ax}\|_p \leq \|\mathbf{Bx}\|_p \leq (1+\epsilon)\|\mathbf{Ax}\|_p$$

in $O(\mathrm{nnz}(\mathbf{A}) \log_R(d) + Rd^\omega \log d)$ time and the number of rows in $\mathbf{B}$ can be bounded by:

$$\begin{cases} O((\alpha\beta R)^p d \log(d) \epsilon^{-2}) & \text{if } 1 \leq p \leq 2 \\ O((\alpha\beta R)^p d^{\frac{p}{2}} \log(d) \epsilon^{-2}) & \text{if } 2 \leq p \end{cases}$$

A. Sampling Using ℓ_2-leverage scores

Our starting point is the observation that a good basis for ℓ_2, specifically a nearly orthonormal basis of $\mathbf{A}$ still allows us to reduce the number of rows substantially under ℓ_p.

Lemma 5.5: If $\mathbf{C} \in \mathbb{R}^{d \times r}$ satisfies $\frac{1}{2}(\mathbf{A}^T\mathbf{A})^\dagger \preceq \mathbf{CC}^T \preceq \frac{3}{2}(\mathbf{A}^T\mathbf{A})^\dagger$ then $\mathbf{U} = \mathbf{AC}$ is a (α, β, p)-well-conditioned basis for $\mathbf{A}$ where $\alpha\beta \leq O(n^{|\frac{1}{2}-\frac{1}{p}|} d^{|\frac{1}{2}-\frac{1}{p}|+\frac{1}{2}})$.

One way to generate such a nearly-orthonormal basis is by the L_2 approximation that we computed in Section IV. This leads to a fast algorithm, but the dependency on n in this bound precludes a single application of sampling using the values given because each time we transfer n rows into $O(p|\frac{1}{2} - \frac{1}{p}|)$ rows. However, note that when $p < 4$, $p|\frac{1}{2} - \frac{1}{p}| = |1 - \frac{p}{2}| < 1$. This means it can be used as a reduction step in an iterative algorithm where the number of rows will decrease geometrically. Therefore, we can use this process as a reduction routine. We will also state the routine to take an approximation $\tilde{\mathbf{A}}$ and compute a basis from it. It gives the following guarantee:

Lemma 5.6: For any constant c, there is an algorithm REDUCEP such that if $\mathbf{A}$ and $\tilde{\mathbf{A}}$ satisfy

$$\frac{1}{2}\|\mathbf{Ax}\|_p \leq \left\|\tilde{\mathbf{A}}\mathbf{x}\right\|_p \leq \frac{3}{2}\|\mathbf{Ax}\|_p \qquad \forall \mathbf{x} \in \mathbb{R}^d$$

and $\tilde{\mathbf{A}}$ has $\tilde{n}$ rows, then REDUCEP$(\mathbf{A}, \tilde{\mathbf{A}}, \epsilon)$ returns in $O(\mathrm{nnz}(\mathbf{A}) \log d + \theta + d^{\omega+\theta} \log d)$ time a matrix $\mathbf{B}$ such that with probability at least $1 - d^{-c}$:

$$(1-\epsilon)\|\mathbf{Ax}\|_p \leq \|\mathbf{Bx}\|_p \leq (1+\epsilon)\|\mathbf{Ax}\|_p \qquad \forall \mathbf{x} \in \mathbb{R}^d$$

And the number of rows in $\mathbf{B}$ can be bounded by:

$$\begin{cases} O(n^{1-\frac{p}{2}} d^{2+\theta} \epsilon^{-2}) & \text{if } 1 \leq p \leq 2 \\ O(n^{\frac{p}{2}-1} d^{\frac{3}{2}p-1+\theta} \epsilon^{-2}) & \text{if } 2 \leq p \end{cases}$$

Iterating this reduction routine with $\tilde{\mathbf{A}} = \mathbf{A}$ gives a way to reduce the row count from n to $\mathrm{poly}(d)$ in $O(\log\log(n/d))$ iterations when $p < 4$. Two issues remain: the approximation errors will accumulate across the iterations, and it's rather difficult (although possible if additional factors of d are lost) to bound the reductions of non-zeros since different rows may have different numbers of them. We will address these two issues systematically before giving our complete algorithm.

The only situation where a large decrease in the number of rows does not significantly decrease the overall number of non-zeros is when most of the non-zeros are in a few rows. A simple way to get around this is to 'bucket' the rows of $\mathbf{A}$ by their number of non-zeros, and compute $\mathrm{poly}(d)$ sized samples of each bucket separately. This incurs an extra factor

of $\log d$ in the final number of rows, but ensures a geometric reduction in problem sizes as we iterate.

The error buildup can in turn be addressed by sampling on the rows of the initial $\mathbf{A}$ using the latest approximation for it $\tilde{\mathbf{A}}$. However, since the algorithm can take up to $O(\log d)$ iterations, we need to perform this on a reduced version of $\mathbf{A}$ instead to obtain a $O(\text{nnz}(\mathbf{A}))$ running time. Pseudocode of our algorithm for a single partition where the number of non-zeros in each row are within a constant factor of each other is given in Algorithm 2.

Algorithm 2 Algorithm for Producing Row Sample of Size $\text{poly}(d)$ that Preserves ℓ_p norm

ROWSAMPLEP($\mathbf{A}, p, \epsilon, \delta$)

Input: $n \times d$ matrix $\mathbf{A}$, p, error parameter ϵ, failure probability $\delta = d^{-c}$

Output: Matrix $\mathbf{B}$

if $1 \leq p \leq 2$ **then**
  $n^* \leftarrow O(d^{\frac{4}{p}+\theta} \log^{\frac{2}{p}} d)$
else
  $n^* \leftarrow O(d^{\frac{3}{2}p-1+\theta} \log^{\frac{3p-2}{4-p}} d)$
end if
if $\mathbf{A}$ has n^* or fewer rows **then**
  return $\mathbf{A}$
end if
$\tilde{\mathbf{A}}_0, \tilde{\mathbf{A}} \leftarrow$ REDUCEP($\mathbf{A}, \mathbf{A}, 1/5$)
while $\tilde{\mathbf{A}}$ has more than $\bar{n}$ rows **do**
  $\tilde{\mathbf{A}} \leftarrow$ REDUCEP($\tilde{\mathbf{A}}_0, \tilde{\mathbf{A}}, 1/5$)
end while
$\mathbf{B} \leftarrow$ REDUCEP($\mathbf{A}, \tilde{\mathbf{A}}, \epsilon/2$)
return $\mathbf{B}$

Lemma 5.7: For any c, there is a setting of constants in ROWSAMPLE such that given a matrix $\mathbf{A}$ where each row has between $[s, 2s]$ nonzeros and $p < 4$. ROWSAMPLEP($\mathbf{A}, p, \epsilon$) with probability $1 - d^{-c}$ returns in $O(\text{nnz}(\mathbf{A}) + d^{\omega} \log d)$ time a matrix $\mathbf{B}$ such that:

$$(1-\epsilon) \|\mathbf{Ax}\|_p \leq \|\mathbf{Bx}\|_p \leq (1+\epsilon) \|\mathbf{Ax}\|_p$$

And the number of rows in $\mathbf{B}$ can be bounded by:

$$\begin{cases} O(d^{\frac{4}{p}+\theta}\epsilon^{-2}) & \text{if } 1 \leq p \leq 2 \\ O(d^{\frac{3}{2}p-1+\theta}\epsilon^{-2}) & \text{if } 2 \leq p \leq 4 \end{cases}$$

B. Fewer Rows by Iterating Again

A closer look at the proof of Lemma 5.7 shows that a significant increase in the number of rows comes from dividing by the $1 - |1 - \frac{p}{2}|$ term in the exponent of n. As a result, the row count can be further reduced if the leverage scores are computed using a $\ell_{p'}$ norm approximation where p' is between 2 and p. We need the following generalization of Lemma 5.5.

Lemma 5.8: If $\mathbf{A}$ has rank r, $1 \leq p \leq p' \leq 2$, and $\tilde{\mathbf{A}}$ is a matrix with $\tilde{n}$ rows such that for all vectors $\mathbf{x}$ we have $\frac{1}{2}\|\mathbf{Ax}\|_q \leq \left\|\tilde{\mathbf{A}}\mathbf{x}\right\|_q \leq \frac{3}{2}\|\mathbf{Ax}\|_q$, and $\mathbf{C}$ is a $d \times r$ matrix such that $\frac{1}{2}(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{\dagger} \preceq \mathbf{CC}^T \preceq \frac{3}{2}(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{\dagger}$, then $\mathbf{U} = \mathbf{AC}$ is a (α, β, p)-well-conditioned basis for $\mathbf{A}$ where $\alpha\beta \leq O(n^{\frac{1}{p}-\frac{1}{p'}}\tilde{n}^{\frac{1}{p'}-\frac{1}{2}}d^{\frac{1}{p}})$.

This allows us to compute leverage scores using a $\ell_{p'}$ norm approximation. By Lemma 5.7, such a matrix has $\tilde{n} = O(d^{\frac{4}{p'}}\log^{\frac{2}{p'}} d)$ rows. By Lemma 5.4, the resulting number of rows can be bounded by:

$$O\left(\left(n^{\frac{1}{p}-\frac{1}{p'}}\tilde{n}^{\frac{1}{p'}-\frac{1}{2}}d^{\frac{1}{p}}R\right)^p d\log d\right)$$
$$= O\left(n^{1-\frac{p}{p'}}\left(d^{\frac{4}{p'}}\log^{\frac{2}{p'}} d\right)^{\frac{p}{p'}-\frac{p}{2}} R^p d^2 \log^{\frac{2p}{p'}+1} d\right)$$

This leads to a result analogous to Lemma 5.6. Solving for the fixed point of this process gives the optimized bounds stated in Theorem 2.2.

This method can be used to reduce the number of rows for all values of $1 \leq p \leq 2$. A calculation similar to the above proof leads to a row count of $O(d^{\sqrt{\frac{8}{p}}-2})$. However, using three or more steps does not lead to a significantly better bound since we can only obtain samples with about d rows when $p = 2$. For $p \geq 4$, multiple steps of this approach also allows us to compute $\text{poly}(d)$ sized samples for any value of p. We omit this extension as it leads to a significantly higher row count.

ACKNOWLEDGMENT

We would like to thank Jelani Nelson for helpful discussions. This work is partially supported by the National Science Foundation under grant number CCF-1018463. Richard Peng is supported by a Microsoft Research PhD Fellowship.

REFERENCES

[1] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney, "Sampling algorithms and coresets for ℓ_p regression," *SIAM J. Comput.*, vol. 38, no. 5, pp. 2060–2078, 2009.

[2] M. Magdon-Ismail, "Row sampling for matrix algorithms via a non-commutative Bernstein bound," *CoRR*, vol. abs/1008.0587, 2010.

[3] C. Sohler and D. P. Woodruff, "Subspace embeddings for the l1-norm with applications," in *Proceedings of the 43rd annual ACM symposium on Theory of computing*, ser. STOC '11. New York, NY, USA: ACM, 2011, pp. 755–764.

[4] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *ICML*, 2012.

[5] K. L. Clarkson, P. Drineas, M. Magdon-Ismail, M. W. Mahoney, X. Meng, and D. P. Woodruff, "The fast cauchy transform and faster robust linear regression," in *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '13. SIAM, 2013, pp. 466–477.

[6] K. L. Clarkson and D. P. Woodruff, "Low rank approximation and regression in input sparsity time," *CoRR*, vol. abs/0911.0547, 2012.

[7] M. W. Mahoney and X. Meng, "Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression," *CoRR*, vol. abs/1210.3135, 2012.

[8] J. Nelson and H. L. Nguyen, "Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings," *CoRR*, vol. abs/1211.1002, 2012.

[9] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1993.

[10] J. Candes, E, "Compressive sampling," *Proceedings of the International Congress of Mathematicians*, 2006. [Online]. Available: http://www-stat.stanford.edu/c̃andes/papers/CompressiveSampling.pdf

[11] P. M. Vaidya, "Speeding-up linear programming using fast matrix multiplication," in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1989, pp. 332–337.

[12] V. V. Williams, "Multiplying matrices faster than coppersmith-winograd," in *Proceedings of the 44th symposium on Theory of Computing*, ser. STOC '12. New York, NY, USA: ACM, 2012, pp. 887–898.

[13] Y. Nesterov, "Gradient methods for minimizing composite objective function," Universit catholique de Louvain, Center for Operations Research and Econometrics (CORE), CORE Discussion Papers 2007076, 2007.

[14] K. L. Clarkson, E. Hazan, and D. P. Woodruff, "Sublinear optimization for machine learning," *J. ACM*, vol. 59, no. 5, p. 23, 2012.

[15] F. Song, H. Ltaief, B. Hadri, and J. Dongarra, "Scalable tile communication-avoiding QR factorization on multicore cluster systems," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–11.

[16] E. Agullo, C. Coti, J. Dongarra, T. Herault, and J. Langou, "QR Factorization of Tall and Skinny Matrices in a Grid Computing Environment," in *24th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2010)*, Atlanta, États-Unis, 2010.

[17] P. G. Constantine and D. F. Gleich, "Tall and skinny QR factorizations in mapreduce architectures," in *Proceedings of the second international workshop on MapReduce and its applications*, ser. MapReduce '11. New York, NY, USA: ACM, 2011, pp. 43–50.

[18] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast monte carlo algorithms for matrices i: Approximating matrix multiplication," *SIAM J. Comput.*, vol. 36, no. 1, pp. 132–157, Jul. 2006.

[19] ——, "Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix," *SIAM J. Comput.*, vol. 36, no. 1, pp. 158–183, Jul. 2006.

[20] ——, "Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition," *SIAM J. Comput.*, vol. 36, no. 1, pp. 184–206, Jul. 2006.

[21] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Sampling algorithms for l2 regression and applications," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, ser. SODA '06. New York, NY, USA: ACM, 2006, pp. 1127–1136.

[22] H. Avron, P. Maymounkov, and S. Toledo, "Blendenpik: Supercharging lapack's least-squares solver," *SIAM J. Scientific Computing*, pp. 1217–1236, 2010.

[23] I. Koutis, A. Levin, and R. Peng, "Improved Spectral Sparsification and Numerical Algorithms for SDD Matrices," in *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), C. Dürr and T. Wilke, Eds., vol. 14. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 266–277.

[24] M. Li, G. L. Miller, and R. Peng, "Iterative row sampling," *CoRR*, vol. abs/1211.2713, 2012.

[25] J. D. Batson, D. A. Spielman, and N. Srivastava, "Twice-Ramanujan sparsifiers," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, 2009, pp. 255–262.

[26] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near optimal column-based matrix reconstruction," in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, 2011, pp. 305–314.

[27] R. Ahlswede and A. Winter, "Strong converse for identification via quantum channels," *IEEE Transactions on Information Theory*, vol. 48, no. 3, pp. 569–579, 2002.

[28] M. Rudelson and R. Vershynin, "Sampling from large matrices: An approach through geometric functional analysis," *J. ACM*, vol. 54, no. 4, p. 21, 2007.

[29] R. Vershynin, "A note on sums of independent random matrices after ahlswede-winter," 2009.

[30] N. Harvey, "C&O 750: Randomized algorithms, winter 2011, lecture 11 notes," 2011.

[31] M. Bādoiu, S. Har-Peled, and P. Indyk, "Approximate clustering via core-sets," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, ser. STOC '02. New York, NY, USA: ACM, 2002, pp. 250–257.

[32] P. K. Agarwal, S. Har-peled, and K. R. Varadarajan, "Geometric approximation via coresets," in *Combinatorial and Computational Geometry, MSRI*. University Press, 2005, pp. 1–30.

[33] I. C. F. Ipsen and T. Wentworth, "The Effect of Coherence on Sampling from Matrices with Orthonormal Columns, and Preconditioned Least Squares Problems," *CoRR*, vol. abs/1203.4809, 2012.

[34] J. A. Tropp, "User-friendly tail bounds for sums of random matrices," *Found. Comput. Math.*, vol. 12, no. 4, pp. 389–434, Aug. 2012.

[35] K. L. Clarkson, "Subgradient and sampling algorithms for l1 regression," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '05. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2005, pp. 257–266.

[36] H. Avron and S. Toledo, "Effective stiffness: Generalizing effective resistance sampling to finite element matrices," *CoRR*, vol. abs/cs/1110.4437, 2011.

[37] P. Drineas and M. W. Mahoney, "Effective resistances, statistical leverage, and applications to linear equation solving," 2010.

[38] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, "Faster least squares approximation," *Numer. Math.*, vol. 117, no. 2, pp. 219–249, Feb. 2011.

[39] D. A. Spielman and J. Woo, "A note on preconditioning by low-stretch spanning trees," *CoRR*, vol. abs/0903.2816, 2009.

[40] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, 2008, pp. 563–568.

[41] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of johnson and lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[42] P. Indyk, "Stable distributions, pseudorandom generators, embeddings, and data stream computation," *J. ACM*, vol. 53, no. 3, pp. 307–323, May 2006.