Quantum-Inspired Algorithms from Randomized Numerical Linear Algebra

Nadiia Chepurko
MIT
nadiia@mit.edu

Kenneth L. Clarkson
IBM Research
klclarks@us.ibm.com

Lior Horesh
IBM Research
lhoresh@us.ibm.com

David P. Woodruff
Carnegie Mellon University
dwoodruf@cs.cmu.edu

Abstract

We create classical (non-quantum) dynamic data structures supporting queries for recommender systems and least-squares regression that are comparable to their quantum analogues. De-quantizing such algorithms has received a flurry of attention in recent years; we obtain sharper bounds for these problems. More significantly, we achieve these improvements by arguing that the previous quantum-inspired algorithms for these problems are doing leverage or ridge-leverage score sampling in disguise; these are powerful and standard techniques in randomized numerical linear algebra. With this recognition, we are able to employ the large body of work in numerical linear algebra to obtain algorithms for these problems that are simpler or faster (or both) than existing approaches.

We also consider static quantum-inspired data structures for the above problems, and obtain close-to-optimal bounds for them. To do this, we introduce a new randomized transform we call the Gaussian Randomized Hadamard Transform (GRHT). It was thought in the numerical linear algebra community that to obtain nearly-optimal bounds for various problems such as rank computation, finding a maximal linearly independent subset of columns, regression, low rank approximation, maximum matching on general graphs and linear matroid union, that one would need to resolve the main open question of Nelson and Nguyen (FOCS, 2013) regarding the logarithmic factors in existing oblivious subspace embeddings. We show how to bypass this question, using our GRHT, and obtain optimal or nearly-optimal bounds for these problems. For the fundamental problems of rank computation and finding a linearly independent subset of columns, our algorithms improve Cheung, Kwok, and Lau (JACM, 2013) and are optimal to within a constant factor and a $\log\log(n)$-factor, respectively. Further, for constant factor regression and low rank approximation we give the first optimal algorithms, for the current matrix multiplication exponent.

1 Introduction

In recent years, quantum algorithms for various problems in numerical linear algebra have been proposed, with applications including least-squares regression and recommender systems [HHL09, LGZ16, RML14, GSLW19, ZFF19, BKL+19, vAG19, LMR14, CD16, BCK15]. Some of these algorithms have the striking property that their running times do not depend on the input size. That is, for a given matrix $A \in \mathbb{R}^{n\times d}$ with $\mathtt{nnz}(A)$ nonzero entries, the running times for these proposed quantum algorithms are at most polylogarithmic in n and d, and polynomial in other independent parameters of A, such as $\mathsf{rank}(A)$, the condition number $\kappa(A)$, or Frobenius norm $\|A\|_F$.

However, as observed by Tang [Tan19] and others, there is a catch: these quantum algorithms depend on a particular input representation of A, which is a simple data structure that allows A to be employed for quick preparation of a quantum state suitable for further quantum computations. This data structure, which is a collection of weight-balanced trees, also supports rapid weighted random sampling of A, for example, sampling the rows of A with probability proportional to their squared Euclidean lengths. So, if an "apples to apples" comparison of quantum to classical computation is to be made, it is reasonable to ask what can be accomplished in the classical realm using the sampling that the given data structure supports.

In this setting, it has recently been shown that sublinear time is sufficient for least-squares regression using a low-rank design matrix A [GLT18, CLW18], for computing a low-rank approximation to input matrix A [Tan19], and for solving ridge regression problems [GST20], using classical (non-quantum) methods, assuming the weight-balanced trees have already been constructed. Further, the results obtained in [Tan19, GLT18, GST20] serve as appropriate comparisons of the power of quantum to classical computing, due to their novel input-output model: data structures are input, then sublinear-time computations are done, yielding data structures as output.

The simple weighted-sampling data structure used in these works to represent the input can be efficiently constructed and stored: it uses $O(\mathtt{nnz}(A))$ space, with a small constant overhead, and requires $O(\mathtt{nnz}(A))$ time to construct, in the static case where the matrix A is given in its entirety, and can support updates and queries to individual entries of A in $O(\log(nd))$ time. However, the existing reported sublinear bounds are high-degree polynomials in the parameters involved: for instance, the sublinear term in the running time for low-rank least-squares regression is $\tilde{O}(\mathsf{rank}(A)^6\|A\|_F^6\kappa(A)^{16}/\varepsilon^6)$; see also more recent work for ridge regression [GST20].

This combination of features raises the following question:

Question 1: Can the sublinear terms in the running time be reduced significantly, in the dynamic and static settings, while preserving the leading order dependence of $O(\mathtt{nnz}(A))$ *(static) and* $O(\log(nd))$ *per update (dynamic)?*

Perhaps a question of greater importance is the connection between quantum-inspired algorithms and the vast body of work in randomized numerical linear algebra: see the surveys [KV09, Mah11, Woo14]. There are a large number of randomized algorithms based on sampling and sketching techniques for problems in linear algebra, yet prior to our work, none of the quantum-inspired algorithms, which are sampling-based, have even mentioned the word "leverage score", for example, which is a powerful tool in randomized numerical linear algebra.

Question 2: Can the large body of work in randomized numerical linear algebra be applied effectively in the setting of quantum-inspired algorithms?

1.1 Our Results

We answer both of the questions above affirmatively. In fact, we answer Question 1 by answering Question 2. Namely, we obtain significant improvements in the sublinear terms for the dynamic (fast updates to the input matrix) and static settings, and our analysis relies on simulating *leverage score sampling* and *ridge leverage score sampling*, using the mentioned weight-balanced data structure that samples rows proportional to squared Euclidean norm.

1.1.1 Connection to Classical Linear Algebra and the Dynamic Case.

The work on quantum-inspired algorithms builds data structures for sampling according to the squared row and column lengths of a matrix. This is also a common technique in randomized numerical linear algebra - see the recent survey on length-squared sampling by Kannan and Vempala [KV17]. However, it is well-known that leverage score sampling often gives stronger guarantees than length-squared sampling; leverage score sampling was pioneered in the algorithms community in [DMM06], and made efficient in [DMIMW12] (see also analogous prior work in the ℓ_1 setting [Cla05]).

Given an $n \times d$ matrix A, with $n \geqslant d$, its (row) leverage scores are the squared row norms of U, where U is an orthonormal basis with the same column span as A. One can show that any choice of basis gives the same scores. Writing $A = U\Sigma V^T$ in its thin singular value decomposition (SVD), and letting A_i and U_i denote the i-th rows of A and U respectively, we see that $\|A_i\|_2^2 = \|U_i\Sigma\|_2^2$, and consequently, $\|A_i\|_2^2 \geqslant \|U_i\|_2^2\sigma_{min}^2(A)$, and $\|A_i\|_2^2 \leqslant \|U_i\|_2^2\sigma_{max}^2(A)$, where $\sigma_{max}(A)$ and $\sigma_{min}(A)$ are the maximum and minimum non-zero singular values of A.

Thus, sampling according to the squared row norms of A is equivalent to sampling from a distribution with ratio distance at most $\kappa^2(A) = \frac{\sigma_{max}(A)^2}{\sigma_{min}(A)^2}$ from the leverage score distribution. This is crucial, as it implies using standard arguments (see, e.g., [Woo14] for a survey) that if we oversample by a factor of $\kappa^2(A)$, then we obtain the same guarantees for various problems that leverage score sampling achieves. But the running times of quantum-inspired algorithms, e.g., the aforementioned $\tilde{O}(\mathsf{rank}(A)^6\|A\|_F^6\kappa(A)^{16}/\varepsilon^6)$ time for regression of [GLT18] and the $\tilde{O}(\|A\|_F^8\kappa(A)^2/(\sigma_{min}(A)^6\varepsilon^4))$ time for regression of [GST20], both take a number of squared-length samples of A depending on $\kappa(A)$, and thus are implicitly doing leverage score sampling, or in the case of ridge regression, ridge leverage score sampling.

Given the connection above, for regression, ridge regression, and low rank approximation in the dynamic case, we show how to obtain simpler algorithms or arguments than those in the quantum-inspired literature by using existing approximate matrix product and subspace embedding guarantees of leverage score sampling. In some sense, this de-mystifies what the rather involved ℓ_2-sampling arguments of quantum-inspired work are doing. We also obtain simpler algorithms and/or tighter bounds than in previous work.

We now describe our concrete results for dynamic data structures in more detail. Our results are summarized in Table 1. Our algorithm for ridge regression, Algorithm 11, does the following: collect a subset of the rows of A via length-squared sampling; take a length-squared sample of the columns of that subset, and then solve a linear system on the resulting small submatrix using the conjugate gradient method. Our analysis of this algorithm is the following theorem. (In the theorem, and this paper, a *row sampling matrix* S has rows that are multiples of natural basis vectors, so that SA is a (weighted) sample of the rows of A. A column sampling matrix is similar.)

Table 1: New results and prior work, dynamic case; notation: error ε, target rank k, $\psi_\lambda \overset{def}{=} \|A\|_F^2/(\lambda+\hat{\sigma}_k^2)$, $\psi_k \overset{def}{=} \|A\|_F^2/\sigma_k(A)^2$, $\kappa_\lambda^2 \overset{def}{=} (\lambda+\sigma_1^2(A))/(\lambda+\sigma_k^2(A))$, $\hat{\kappa}^2 \overset{def}{=} (\lambda+\hat{\sigma}_1^2)/(\lambda+\hat{\sigma}_k^2)$, where $\hat{\sigma}_k \leqslant 1/\|A^+\|$, $\hat{\sigma}_1 \geqslant \|A\|$, d' is the number of columns of B for multiple-response, η denotes some numerical properties of A.

Problem	Time		Prior Work	
	Update	Query	Update	Query
Least-Squares Regression	$O(\log(n))$	$\tilde{O}(d'\varepsilon^{-4}\hat{\kappa}^2\psi_\lambda^2\kappa\log(d))$ Thm. 52	$O(\log(n)$	$\tilde{O}(\frac{k^6\|A\|_F^6\kappa^{16}}{\varepsilon^6})$ [GLT18] $\tilde{O}(\frac{\|A\|_F^8\kappa(A)^2}{(\sigma_{\min}^6\varepsilon^4)})$ [GST20]
Low Rank Sampling	$O(\log(n))$	$\tilde{O}(\varepsilon^{-4}\psi_\lambda(\psi_\lambda+k^2+k\psi_k)+\varepsilon^{-6}k^3)$ Thm. 57	$O\log(n)$	$\Omega(\text{poly}(\eta k\varepsilon^{-1}))$ [Tan19]

Theorem 1 (Entries from Least-Squares Regression, informal Theorem 52) *Given an $n\times d$ matrix A for which a sampling data structure has been maintained, a $n\times d'$ matrix B, error parameter $\varepsilon>0$, and ridge parameter λ, let X^* be the optimal ridge regression solution, i.e. $X^* \overset{def}{=} \operatorname{argmin}_X \|AX-B\|_F^2+\lambda\|X\|_F^2$. Then, a sketching matrix $S\in\mathbb{R}^{m_s\times n}$ that samples rows of A, and and approximate solution $\tilde{X}\in\mathbb{R}^{m_s\times d'}$, can be found such that*

$$\|A^\top S^\top\tilde{X}-X^*\|_F \leqslant \varepsilon\left(\|X^*\|_F\gamma^2+\frac{1}{\sqrt{\lambda}}\|U_{\lambda,\perp}B\|_F\right),$$

where $U_{\lambda,\perp}B$ is the projection of B on to the subspace corresponding to the singular values of SA less than λ; and γ is a problem dependent parameter ($\gamma=\frac{\|B\|_F}{\|AA^+B\|_F}$ when $\lambda=0$).

Further, the running time to find $\tilde{X}$ is $\tilde{O}(d'\varepsilon^{-4}\psi_\lambda^2\kappa_\lambda\log(d))$, where κ_λ is the ridge condition number of A, and $\psi_\lambda \overset{def}{=} \|A\|_F^2/(\lambda+\hat{\sigma}_k^2)$, with $\hat{\sigma}_k$ a lower bound $\hat{\sigma}_k\leqslant\sigma_{rank(A)}(A)$. Finally, for all $i\in[d]$, and $j\in[d']$, an entry $(A^\top S^\top\tilde{X})_{i,j}$ can be found in $O(m_S\log(nd))=O(\varepsilon^{-2}\kappa_\lambda^2(A)\psi_\lambda(\log(nd))^2)$ time.

The "numerical" quantities κ_λ, and ψ_λ are decreasing in λ, and $\frac{1}{\sqrt{\lambda}}\|U_{k,\perp}B\|_F$ is plausibly roughly at most $\|AA^+B\|_F/\|A\|$. When λ is within a constant factor of $\|A\|^2$, ψ_λ is close to the *stable rank* $\|A\|_F^2/\|A\|^2$, where the stable rank is always at most $\text{rank}(A)$.

Concurrent Work. In an independent and concurrent work, Gilyén, Song and Tang [GST20] obtain a roughly comparable classical algorithm for regression, assuming access to the weight balanced tree data structures, which runs in time $\tilde{O}\left(\frac{\|A\|_F^6\|A\|_2^2}{\|A^+\|_2^8\varepsilon^4}\right)$, or in the notation above, $\tilde{O}(\varepsilon^{-4}\psi_\lambda^3\kappa^2)$. Their algorithm is based on Stochastic Gradient Descent and the $\text{nnz}(A)$ term in the running time of their algorithm does not get multiplied by $\mathrm{poly}(1/\varepsilon)$ factors, despite the $O^*(\cdot)$ notation used in that paper.

We also give dynamic Algorithm 12 for approximating A with a rank-k matrix, for given k, and a means of sampling from it in the mode of [Tan19]. As with our ridge regression algorithm, we first length-squared sample rows and columns; then sample the resulting submatrix to one with $\tilde{O}(\varepsilon^{-2}k)$ rows and columns; compute the SVD of that matrix; and then work back up to A with more sampling and a QR factorization. Our algorithm and analysis rely on *Projection-Cost Preserving* sketches, as discussed also in Section 3.1. We have the following.

Theorem 2 (Sampling from a low-rank approximation, informal Theorem 57) *Given an $n \times d$ matrix A for which a sampling data structure has been maintained, target rank k, error parameter ε, and estimates of numerical properties of A (such as $\|A - A_k\|_F^2$, the error of the best rank-k approximation A_k), we can find sampling matrices S and R, and rank-k matrix W, such that*

$$\|ARWSA - A\|_F \leqslant (1 + O(\varepsilon))\|A - A_k\|_F.$$

Further, the running time is $\tilde{O}(\varepsilon^{-6}k^3 + \varepsilon^{-4}\psi_\lambda(\psi_\lambda + k^2 + k\psi_k))$, where ψ_λ is as in Theorem 1, and ψ_k is an estimate of $\|A\|_F^2/\sigma_k(A)^2$. Given $j \in [d]$, a random index $i \in [n]$ with probability distribution $(ARWSA)_{ij}^2/\|ARWSA)_{*,j}\|^2$ can be generated in expected time $\tilde{O}(\psi_k + k^2\varepsilon^{-2}\kappa(A)^2)$, where ψ_k is an estimate of $\|A\|_F^2\|A^+\|_2^2$ and $\kappa \overset{def}{=} \|A\|\|A^+\|$.

This result is directly comparable to Tang's algorithm [Tan19] for recommender systems which again needs query time that is a large polynomial in k, κ and ε^{-1}. Our algorithm returns a relative error approximation, a rank-k approximation within $1+\varepsilon$ of the best rank-k approximation; Tang's algorithm has additive error, with a bound more like $\|A - A_k\|_F + \varepsilon\|A\|_F$.

1.1.2 Static Algorithms.

We also obtain new results for static data structures, in many cases removing, in particular, the *last* log factor in a running time dependence. We note that the bottleneck in improving prior work, including such removal of last logs, involved well-known conjectures to construct *Sparse Johnson-Lindenstrauss transforms* (see Conjecture 14 in [NN13]).

To side-step these conjectures we introduce a new simple matrix sketching technique: multiplication by a random sparse matrix whose randomly chosen nonzero entries are Gaussians. We show that by composing this matrix with a Subsampled Randomized Hadamard Transform (SRHT), a sketching scheme is obtained that allows smaller (by a log factor) sketching dimension, while requiring the same sketching time as SRHT. This scheme, which we call the *Gaussian Randomized Hadamard Transform*, or GRHT, thus constitutes a "plugin replacement" for the SRHT, removing a log factor that has thus far remained both a nuisance and an impediment to optimal algorithms.

Using the techniques we introduce in this work, we obtain nearly-optimal (up to $\log\log$ factors in the sublinear terms) running time for fundamental problems in classical linear algebra and graph algorithms, including computing matrix rank, finding a set of independent rows, linear regression, maximum matching and linear matroid union. Further, for regression (in a low precision regime) and low-rank approximation, we obtain the first optimal algorithms for the current matrix multiplication exponent.

We now describe our results in more detail. Table 2 summarizes our results. We begin with least-squares regression:

Theorem 3 (Least-Squares Regression, informal Theorem 49) *Given an $n \times d$ matrix A with $k \overset{def}{=} rank(A)$, and a vector b, there exists an algorithm that computes $\hat{x}$ such that $\|A\hat{x} - b\|_2 \leqslant (1+\varepsilon)\min_x \|Ax - b\|_2$ in time $O(\texttt{nnz}(A) + \min(d^\omega, k^\omega \log\log(n)) + k^\omega/\varepsilon)$. Here multiplication of $k \times k$ matrices needs $O(k^\omega)$ time.*

We note that for constant ε, the running time obtained is within a $\log\log(n)$ factor of optimal, for the current matrix multiplication constant. Further, it improves on prior work by Clarkson and Woodruff [CW13], who obtain an algorithm with additional $\log(n)$ factor multiplying the leading $\texttt{nnz}(A)$ term. Next, we show a similar result holds for low-rank approximation:

Theorem 4 (LRA in Current Matrix Multiplication Time, informal Theorem 37) *Given $\varepsilon > 0$, a $n \times d$ matrix A and $k \in [n]$, there exists an algorithm that runs in $O\left(\texttt{nnz}(A) + \frac{dk^{\omega-1}}{\varepsilon} + \frac{dk^{1.01}\log^c(k)}{\varepsilon^2}\right)$ time and outputs a $d \times k$ matrix Q with orthonormal columns such that with probability $9/10$, $\|A - AQQ^T\|_F^2 \leqslant (1+\varepsilon)\|A - A_k\|_F^2$.*

Our results use the following leverage score sampling primitive, which may be of independent interest:

Theorem 5 (Faster Leverage Score Sampling, informal Theorem 45) *Given an $n \times d$ matrix A such that $k = rank(A)$, there exists an algorithm to sample $k\log(k)/\varepsilon^2$ rows proportional to ε-approximate leverage scores in $O(\texttt{nnz}(A) + k^\omega \log\log(n) + k^2\varepsilon^{-2.1})$ time and $O(n + k^\omega \log\log(n))$ space.*

Finally, we obtain faster algorithms for computing the rank of a matrix and finding a full-rank set of rows.

Theorem 6 (Matrix Rank and Finding a Basis, informal Theorem 32 and 33) *Given an $n \times d$ matrix A, there exists a randomized algorithm to compute $k = rank(A)$ in $O(\texttt{nnz}(A) + k^\omega)$ time and $O(n + k^{2.1})$ space, where ω is the matrix multiplication constant. Further, the algorithm can find a set of k linearly independent rows in $O(\texttt{nnz}(A) + k^\omega \log\log(n))$ time and $O(n + k^\omega \log\log(n))$ space.*

We note that this result improves prior work by Cheung, Kwok and Lau [CKL13], who obtain a $O(\texttt{nnz}(A)\log(n) + k^\omega)$ time algorithm to compute matrix rank and $O(\log(n)(\texttt{nnz}(A) + k^\omega))$ time algorithm to find a full-rank set of rows.

1.2 Our Techniques

We provide a further overview of our results and techniques below, for our static algorithms.

1.2.1 Least-Squares Regression

We have already discussed our least-squares algorithm for the dynamic case.

We begin with obtaining an approximate optimizer for the least-squares regression problem. Our setting is in general similar to that of Gilyén, Lloyd and Tang [GLT18]. We first find $k =$

Table 2: New results and prior work, static case; space not including storage of A. Parameters: k is target rank for low-rank approximation; $k = \mathsf{rank}(A)$; κ is the condition number of A; $\nu =$ number of samples; $\mu_s > 0$ a tunable parameter; multiplying two $k \times k$ matrices takes $O(k^\omega)$ time.

Problem	Time	Prior Time	Space
Rank Computation	$O(\mathtt{nnz}(A) + k^\omega)$ Thm. 32	$O(\mathtt{nnz}(A) \log d + k^\omega)$ [CKL13]	$O(n + k^{2.01})$
Finding a Basis	$O(\mathtt{nnz}(A)+$ $k^\omega \log\log(n))$ Thm. 33	$O(\mathtt{nnz}(A) \log d$ $+k^\omega \log d)$ [CKL13]	$O(n + k^\omega \log\log(n))$
Low Rank Approximation	$O(\mathtt{nnz}(A) + \varepsilon^{-1} dk^{\omega-1}$ $+\varepsilon^{-2} dk^{1.01})$ Thm. 37	$\Omega(\mathtt{nnz}(A) \log(d))$	$O(\varepsilon^{-2} dk^{1.01}))$
Leverage Score Sampling	$O(\mathtt{nnz}(A) + k^\omega \log\log(n)$ $+k^2 \log(n) + \nu k n^{1/100})$ Thm. 45	$\Omega(\mathtt{nnz}(A))$ see text	$O(n + k^\omega \log\log(n))$
Least Squares Iterative	$O((\mathtt{nnz}(A) + k^2) \log(\kappa/\varepsilon)$ $+k^\omega \log\log(n))$ Thm. 46	$\Omega(\mathtt{nnz}(A) \log d)$ [CW13]	$O(n + k^\omega \log\log(n))$
Least Squares Residual	$O(\mathtt{nnz}(A) + d^\omega \varepsilon^{-1}$ $+d^2 \log(n))$ Thm. 49	$\Omega(\mathtt{nnz}(A) + d^{\omega+0.1} \varepsilon^{-2})$ [MP12]	$O(n + k^\omega \log\log(n))$
Least Squares Multi-Response	$O(\mathtt{nnz}(A) + k^2 \log(n)$ $+k^\omega \varepsilon^{-2.01}$ Thm. 48	NA	$O(n + k^\omega \log\log(n))$
Low-Rank Sampler	$O(\mu_s \,\mathtt{nnz}(A))$ $+\tilde{O}(\mu_s k^3 (k + \varepsilon^{-7}))$ $+O(\nu(\varepsilon^{-1} k + k^2)^2 n^{1/\mu_s})$ Thm. 55	NA	$O(\mathtt{nnz}(A)$ $+\tilde{O}(k^2(k + \varepsilon^{-1}))$

$\mathsf{rank}(A)$ linearly independent columns of A (also called *finding a basis*) and proceed using only those columns. Let the corresponding matrix be $A\Lambda$, where $\Lambda \in \mathbb{R}^{d \times k}$, whose columns are a subset of the $d \times d$ identity matrix, is such that $\mathsf{rank}(A\Lambda) = \mathsf{rank}(A)$. We then solve the regression problem using $A\Lambda$ as a proxy for A, and while this yields a *different* optimal solution (x^*) perhaps, than without such reduction, it has the same prediction quality. Moreover, after finding a basis, it is no longer necessary to output a data structure to allow sublinear work, since the output vector will have k nonzero entries.

Following this reduction, for our first least-squares algorithm, we quickly find a pre-conditioner, then solve the pre-conditioned version of the problem iteratively, so that the dependence on the condition number $\kappa(A)$ and error parameter ε is logarithmic. Overall, this algorithm requires $O(\mathtt{nnz}(A) \log(\kappa(A)/\varepsilon))$ time and improves the algorithm appearing in [CW13], since the basis-finding subroutine can now be implemented without a $\log(n)$ overhead (see Theorem 46 for a formal statement). We provide an overview of the *basis finding* algorithm in Subsection 1.2.4.

While our first algorithm for least-squares is fast, it touches all the input data multiple times, and in particular does so even if the design matrix A is fixed, and multiple response vectors b are presented. In contrast, the algorithm of [GLT18] runs in time independent of $\mathtt{nnz}(A)$, assuming the input data structures are already built. So in the situation where there is one design matrix A, but many response (right-hand-side) vectors b, this could potentially be faster than our first algorithm.

However, Theorem 38 of [CW13] gives an algorithm for *multiple-response* regression (called there *generalized* regression), where in addition to A, there is input $B \in \mathbb{R}^{n\times d'}$, and the goal is to find a solution with low relative prediction error. That is, for $X^* = \text{argmin}_{X\in\mathbb{R}^{n\times d'}}\|AX - B\|_F$, Theorem 38 of [CW13] shows that there is a sampling matrix S so that for

$$\tilde{X} = \text{argmin}_{X\in\mathbb{R}^{n\times d'}}\|S(AX - B)\|_F$$

with constant probability, $\|A\tilde{X} - B\|_F \leqslant (1+\varepsilon)\|AX^* - B\|_F$. The sampling matrix does not depend on B, so that, assuming the columns of B are not chosen *adaptively*, or with knowledge of S, the given bounds will hold in a setting where the columns of B are presented sequentially.

We adapt this approach here, for another algorithm, applied to a setting where we are interested in estimating X^*, not predicting B, and where space is of particular interest. The "query" time in Table 2 is the time to compute one column of our estimate $\tilde{X}$, given a column of B. In the interests of requiring small space, we use the technique of leverage-score sampling. This allows the selection and use of $O(\text{poly}(k/\varepsilon))$ rows of A, allowing estimates with ε error. Our error bounds are of the form $\|\tilde{X} - X^*\|_F^2 \leqslant \varepsilon\|A\|\|AX^* - B\|_F$; this can be related to the bounds of [GLT18] using the assumptions they use relating $\|Ax^*\|$ to $\|Ax^* - b\|$, here, per-column of X^* and B.

By applying these ideas and the Gaussian Randomized Hadamard Transform (GRHT), we can push the time for least-squares regression in the setting of minimizing residual error down to near-optimal, as shown in the table. The formal statement, Theorem 49, gives a bound for multiple-response regression, and first selects $\text{rank}(A)$ basis columns, so that the running time can be expressed in terms of $\text{rank}(A)$.

1.2.2 Sampling from a Low-Rank Approximation

In Section 8, we give data structures analogous to that of [Tan19] for recommender systems, supporting sampling from a given row (or equivalently, column) of a low-rank approximation to the input matrix, as discussed above. We have already discussed the dynamic case. For the static case, we have the following.

Theorem 7 (Sampling from LRA, informal Theorem 55) *Given an $n \times d$, rank-k matrix A, we can build a data structure of size $O(\text{nnz}(A) + k^3 + k^2/\varepsilon)$ in time $O(\text{nnz}(A) + k^4 + k/\varepsilon^7)$ such that given $j \in [d]$, it outputs an $i \in [n]$ with probability proportional to $\hat{A}_{i,j}^2/\|\hat{A}_{*,j}\|_2^2$, where $\|A - \hat{A}\|_F^2 \leqslant (1+\varepsilon)\|A - A_k\|_F^2$. Further, the expected query time $O((k/\varepsilon + k^2)n^{0.1})$.*

Our data structures use a prior algorithm for finding a good low-rank approximation, using sketches, and then applies our matrix-vector product machinery to support the random generation recommender queries discussed above.

Our algorithm adopts prior machinery to quickly find a "near-SVD" of A, specifically, matrices R, S, $\hat{W}$, $\hat{Z}$, and Σ so that $AR\hat{W}$ and $\hat{Z}SA$ have singular values all close to 1, Σ is a diagonal matrix, and $A \approx \hat{A} \overset{def}{=} AR\hat{W}\Sigma\hat{Z}SA$. This implicit factorization can be found in time $O(\text{nnz}(A)) + \text{poly}(k/\varepsilon)$; we store AR, SA, and the other matrices, in space $O(\text{nnz}(A) + \text{poly}(k/\varepsilon))$. We also need some additional sampling machinery, needing $O(\text{nnz}(A) + \text{poly}(k/\varepsilon))$ time to build. We then use a rejection technique inspired by [Tan19], although here based on leverage-score sampling, so that, given column $w \leftarrow \hat{Z}(SA)_{*,j}$ of the right-hand factor of $\hat{A}$, we can pick row $(AR)_{i,*}\hat{W}$ with probability proportional to $((AR)_{i,*}\hat{W}w)^2$, that is, proportional to $\hat{A}_{ij}^2$.

The query time for the static data structure is $O(\text{poly}(k/\varepsilon)n^{1/\lambda})$, where k is the target rank. note that when $\lambda = \log n$, the $n^{1/\lambda}$ term in the query time is equal to one, although this is at the cost of time $O(\mathtt{nnz}(A)\log n)$ in pre-processing.

An extremal case of some interest here is when $\mathsf{rank}(A) = k$, so that $A = [A]_k$, so that we have also $\hat{A} = A$, since we have bounded relative error.

1.2.3 Leverage-score sampling

Recall that the leverage score τ_i of row $A_{i,*}$ of matrix A is the squared Euclidean norm of $U_{i,*}$, where U is any matrix with orthonormal columns such that $\operatorname{im} A = \operatorname{im} U$; another description is that the row $A_{i,*}$ has leverage score $\tau_i = A_{i,*}(A^\top A)^+ A_{i,*}^\top$. (Here X^+ is the Moore-Penrose pseudo-inverse of matrix X, as discussed in Subsection 2.) Note that $\sum_{i\in[n]} \tau_i = k \overset{def}{=} \mathsf{rank}(A)$. A sketching matrix L based on leverage scores would have each of its rows chosen by picking $i \in [n]$ with probability p_i proportional to τ_i/k; the corresponding row of L would then be $e_i^\top/\sqrt{p_i}$, where e_i is the i'th natural basis vector. As discussed in Lemma 9, such matrices L yield good subspace embeddings for A, and as a result have application in a variety of matrix algorithms, including for regression and low-rank approximation.

Theorem 8 (Faster Leverage Score Sampling, informal Theorem 45) *Given an $n \times d$ matrix A such that $k = \mathit{rank}(A)$, there exists an algorithm to sample $k\log(k)/\varepsilon^2$ rows proportional to ε-approximate leverage scores in $O(\mathtt{nnz}(A) + k^\omega \log\log(n) + k^2\varepsilon^{-2.1})$ time and $O(n + k^\omega \log\log(n))$ space.*

In Section 6, we sharpen prior algorithms for leverage-score sampling, by avoiding the accurate computation of all of the leverage scores. A standard approach for such computation entails the use of a matrix $G \in \mathbb{R}^{n\times m_G}$, where $m_G = \Theta((\log n)/\varepsilon^2)$, such that vector $x \in \mathbb{R}^n$ has $\|x^\top G\| = (1 \pm \varepsilon)\|x\|$ with failure probability $1/n^{\Omega(1)}$. Computing the matrix $A \cdot G$ requires $\Omega(\mathtt{nnz}(A)\log n)$ time, even for constant ε. We show that we can avoid the $\log n$ factor and obtain truly input sparsity time algorithms.

We employ a scheme introduced by Clarkson and Woodruff [CW15] using $m_G = O(1)$ to get *crude* estimates for the leverage scores. These crude estimates could be used directly for sampling, but the cruder the estimate, the larger the necessary sample size relative to the sample size needed using accurate leverage scores. We then use rejection sampling to simulate a sample drawn from *accurate* leverage scores. We thereby avoid a larger sample size, but we do have a mild tradeoff between terms in the runtime bound, similar to other tradeoffs here. This is similar to that achieved by other algorithms for computing leverage scores, by other specific algorithms [LMP13, CNW15, CLM+15].

Matrix-vector product sampling. A key operation for leverage-score sampling, and for recommender queries, is random row generation: given $w \in \mathbb{R}^n$, generate random row index i with probability proportional to $(A_{i,*}w)^2/\|Aw\|^2$. Tang [Tan19] gave a data structure supporting such operations; in Section 5, we give a version for the static case, generalized slightly to allow matrices W so that the probability is $\|A_{i,*}W\|^2/\|AW\|_F^2$.

1.2.4 Rank-related Computations

We begin by considering the fundamental problem of computing the rank of a matrix and closely related problem of computing a set of full-rank subset of rows of a given matrix.

We note that this improves the rank and basis-finding algorithms of Cheung, Kwok and Lau [CKL13], who incur additional $\log(d)$ factors in the running time. Further, it is easy to see that $\Omega(\mathtt{nnz}(A))$ is a lower bound for both these problems. As a direct consequence of our techniques, we obtain a $O(|E| + k^{\omega} \log\log(n))$ algorithm for Maximum Matching and a fast algorithm for Linear Matroid Union (see Section 4.5 for details). Finally, using the techniques developed above, we obtain a *current matrix-multiplication time* algorithm for low-rank approximation (see Theorem 37).

To obtain linear dependence on $\mathtt{nnz}(A)$, we sharpen Cheung, Kwok, and Lau [CKL13] results in Section 4, removing log factors, at the expense of using a bit more space, as shown in Table 2. This improvement is done by taking careful advantage of the relative magnitudes of $\mathtt{nnz}(A)$ and $\mathsf{rank}(A)$: when $\mathsf{rank}(A)$ is small enough, the compression techniques of [CKL13] can be used to shrink the problem to one with at most about $\mathtt{nnz}(A)/\log n$ nonzero entries, so that $O(\mathtt{nnz}(A))$ terms in the time bounds are possible.

For finding a basis, we apply leverage score sampling, accelerated by our Sparse Gaussian sketch, to find $m = k\text{polylog}(k)$ rows that contain k linearly independent rows; we then apply the algorithm of [CKL13], which needs $\log(m/k) = \log\log k$ iterations, each taking k^{ω} time, and thus yields a faster algorithm.

Subspace embeddings and SparseGaussian sketches. Central to the above result and throughout the remaining paper is the machinery of *subspace embeddings*: for matrix A and $\varepsilon > 0$, a subspace ε-embedding (or just ε-embedding) is a matrix S such that for all $x \in \mathbb{R}^n$, $1 - \varepsilon \leqslant \|SAx\|/\|Ax\| \leqslant 1 + \varepsilon$. While several standard constructions are known (see Lemma 9), obtaining the optimal trade-off between sparsity of the sketch and the number of rows required (also known as sketching dimension) remained a central open problem (Conjecture 14 in [NN13]). Count-Sketch matrices were first shown to satisfy the subspace embedding property in [CW13]. Subsequently, this sketch was extended by Nelson and Nguyen [NN13] to OSNAP sketches. The best known analysis by Cohen, based on OSNAP sketches, has sketching dimension $O(d \log(d)/\varepsilon^2)$ and $\log(d)$ non-zeros in each column and thus can be applied to a matrix in $\mathtt{nnz}(A)\log(d)$ time [Coh16], while incurring a linear dependence on d in the sketching dimension. We note that more generally, an algorithm with running time $O(\gamma\,\mathtt{nnz}(A))$ and sketching dimension $d^{1+1/\gamma}$ can be obtained[NN13]. Since subspace embeddings are a fundamental primitive in numerical linear algebra, algorithms for regression, low-rank approximation, matrix rank and other downstream applications incur a $\mathtt{nnz}(A)\log(d)$ running time when the sketching dimension is near-optimal.

In this work, we introduce and analyze the $\mathsf{SparseGaussian}$ sketch, which has $O(d/\varepsilon^2)$ rows and each entry is generated independently, and with probability $\text{polylog}(n)/n$, it is set to be a random Gaussian value, and is otherwise zero. We show that composing such a matrix with the Subsampled Randomized Hadamard Transform (see Lemma 9), we obtain a sketching matrix that is a subspace embedding with $O(d/\varepsilon^2)$ rows (see Theorem 16). Moreover, as with other sketching techniques, this *Gaussian Randomized Hadamard Transform* (GRHT) supports fast matrix product estimation and construction of projection-cost preserving sketches; these are discussed in Subsection 3.1. Finally, composing GRHT with Count Sketch, we obtain subspace embedding that has $O(d/\varepsilon^2)$ sketching dimension and can be computed in $\mathtt{nnz}(A)$ time. This side-steps the main conjecture of [NN13], which was whether countsketch-based subspace embeddings need $\Omega(\mathtt{nnz}(A)\log d)$ sketching time to achieve optimal sketching dimension.

1.3 Summary of Related Work

Matrix Sketching. The *sketch and solve* paradigm [CW15, Woo14] was designed to reduce the dimensionality of a problem, while maintaining enough structure such that a solution to the smaller problem remains an approximate solution the original one. This approach has been pivotal in speeding up basic linear algebra primitives like least-squares regression [Sar06, RT08, CW15], ℓ_p regression[CP15, WW19], low-rank approximation [NN13, CMM17, LW20], linear and semi-definite programming [CLS19, JSWZ20, JKL+20] and solving non-convex optimization problems such as ℓ_p low-rank approximation [SWZ17, SWZ19, BBB+19] and training neural networks [BJW19, BPSW20]. For a comprehensive overview we refer the reader to the aforementioned papers and citations therein. Several applications use rank computation, finding a full rank subset of rows/columns, leverage score sampling and computing subspace embeddings as key algorithmic primitives. In addition to being used as a block box, we believe our techniques will be useful in sharpening bounds for several such applications.

Sublinear Algorithms and Quantum Linear Algebra. Recently, there has been a flurry of work on sublinear time algorithms for structured linear algebra problems [MW17, SW19, BLWZ19, BCJ20] and quantum linear algebra [HHL09, GSLW19, LMR14, KP16, DW20]. The unifying goal of these works is to avoid reading the entire input to solve tasks such as linear system solving, regression and low-rank approximation. The work on sublinear algorithms assumes the input is drawn from special classes of matrices, such as positive semi-definite matrices [MW17, BCW19], distance matrices [BW18, IVWW19] and Toeplitz matrices [LLMM20], whereas the quantum algorithms (and their de-quantized analogues) assume access to data structures that admit efficient sampling [Tan19, GLT18, CGL+20].

The work Gilyén, Lloyd and Tang [GLT18] on low-rank least squares produces a data structure as output: given index $i \in [d] \overset{def}{=} \{1, 2, \dots, d\}$, the data structure returns entry x'_i of $x' \in \mathbb{R}^d$, which is an approximation to the solution x^* of $\min_{x \in \mathbb{R}^d} \|Ax - b\|$, where $b \in \mathbb{R}^n$. The error bound is $\|x' - x^*\| \leqslant \varepsilon \|x^*\|$, for given $\varepsilon > 0$. This requires the condition that $\|Ax^* - b\| / \|Ax^*\|$ is bounded above by a constant. Subsequent work [CGL+20] removes this requirement, and both results obtain data structures that need space polynomial in $\mathsf{rank}(A)$, ε, $\kappa(A)$,[1] and other parameters.

The work [Tan19] also produces a data structure, that supports sampling relevant to the setting of recommender systems: the nonzero entries of the input matrix A are a subset of the entries of a matrix P of, for example, user preferences. An entry $A_{ij} \in [0, 1]$ is one if user j strongly prefers product i, and zero if user j definitely does not like product i. It is assumed that P is well-approximated by a matrix of some small rank k. The goal is to estimate P using A; one way to make that estimate effective, without simply returning all entries of P, is to create a data structure so that given j, a random index i is returned, where i is returned with probability $\hat{a}_{ij}^2 / \|\hat{A}_{*,j}\|^2$.

[1] Throughout, we define $\kappa(A) = \|A\| \|A^+\|$, that is, the ratio of largest to smallest *nonzero* singular values of A, so that, in particular, it will never be infinite or undefined.

Here $\hat{A}_{*,j}$ is the j'th column of $\hat{A}$ (and $\hat{a}_{ij}$ an entry), where $\hat{A}$ is a good rank-k approximation to A, and therefore, under appropriate assumptions, to P. The estimate $\hat{A}$ is regarded as a good approximation if $\|\hat{A} - A\|_F \leqslant (1+\varepsilon)\|A - [A]_k\|_F$, where $[A]_k$ is the matrix of rank k closest to A, in Frobenius norm. Here ε is a given error parameter. As shown in [Tan19], this condition (or indeed, a weaker one) implies that the described sampler is useful in the context of recommender systems.

2 Notation and Basics

Let X^+ denote the Moore-Penrose pseudo-inverse of matrix X, equal to $V\Sigma^{-1}U^\top$ when X has thin SVD $X = U\Sigma V^\top$, so that Σ is a square invertible matrix. We note that

$$X^+ = (X^\top X)^+ X^\top = X^\top (XX^\top)^+ \text{ and } X^+XX^\top = X^\top, \tag{1}$$

provable using the SVDs of X and X^+. Also, if X has full column rank, so that V is square, then X^+ is a left inverse of X, that is $X^+X = I_d$, where d is the number of colums of X. Let $\|X\|$ denote the spectral (operator) norm of X. Let $\kappa(X) \overset{def}{=} \|X^+\|\|X\|$ denote the condition number of X. We write $a \pm b$ to denote the set $\{c \mid |c-a| \leqslant |b|\}$, and $c = a \pm b$ to denote the condition that c is in the set $a \pm b$. Let $[m] \overset{def}{=} \{1, 2, \ldots, m\}$ for integer m.

As mentioned $\mathtt{nnz}(A)$ is the number of nonzero entries of A, and we assume $\mathtt{nnz}(A) \geqslant n$, i.e. there are no rows comprising entirely zeros; it also follows by our assumptions that also $\mathtt{nnz}(A) \geqslant n$. We let $[A]_k$ denote the best rank-k approximation to A. Let $0_{a\times b} \in \mathbb{R}^{a\times b}$ have all entries equal to zero, and similarly $0_a \in \mathbb{R}^a$ denotes the zero vector. Further, for a $n \times d$ matrix A and a subset S of $[n]$, we use the notation $A_{|S}$ to denote the restriction of the rows of A to subset indexed by S. We use Knuth's notation $x \uparrow\uparrow y$ to denote a tower of x's, exponentiated y times. As mentioned, n^ω is the time needed to multiply two $n \times n$ matrices.

Lemma 9 (Known constructions of sketching matrices) *For given matrix $A \in \mathbb{R}^{n\times d}$ with $k = \mathit{rank}(A)$, these constructions give ε_0-embeddings with failure probability $1/k^c$, for given constant c. Here sketching matrix S is an ε_0-embedding if $\|SAx\| = (1 \pm \varepsilon_0)\|Ax\|$, for all $x \in \mathbb{R}^n$.*

- *There is a sketching matrix $T \in \mathbb{R}^{m_T \times n}$ with sketching dimension $m_T = O(\varepsilon_0^{-2}k^{1+1/\mu}\log k)$ such that TA can be computed in $O(\mu\,\mathtt{nnz}(A)/\varepsilon_0)$ time (see e.g. [Coh16]), with μ/ε_0 non-zero entries per row, in this form called here an OSNAP, and in earlier forms called a* countsketch *[CW13] matrix, or* sparse embedding. *The most sparse version $\hat{T} \in \mathbb{R}^{m_{\hat{T}} \times n}$ has $m_{\hat{T}} = O(\varepsilon_0^{-2}k^2)$, with $\hat{T}A$ computable in $O(\mathtt{nnz}(A))$ time; $\hat{T}$ has one nonzero entry per column. A less sparse version $\bar{T}$ of OSNAP has $m_T = O(\varepsilon_0^{-2}k\log(nd))$, $O(\log(nd)/\varepsilon_0)$ entries per column, and failure probability $1/\mathrm{poly}(nd)$.*

- *There is a sketching matrix $H \in \mathbb{R}^{m_H \times n}$ with $m_H = O(\varepsilon_0^{-2}k\log(nk))$ such that HA can be computed in $O(nd\log n)$ time (see e.g. [BG13]). This is called an SRHT (Sampled Randomized Hadamard Transform) matrix. The matrix $H = \hat{H}D$, where the rows of $\hat{H}$ are a subset of the rows of a Hadamard matrix, and D is a diagonal matrix whose diagonal entries are ± 1.*

- *If matrix $L \in \mathbb{R}^{m_L \times n}$ is chosen using leverage score sampling (see Section 6), then there is $m_L = O(\varepsilon_0^{-2}k\log k)$ so that L is an ε_0-embedding, [Rud99, Rec11].*

These embeddings can be composed, so that for example $S = H_S T_S$ is a "two-stage" ε_0-embedding for A, where T_S is a OSNAP matrix, and H_S is an SRHT, so that $H_S T_S A$ can be computed in $O(\varepsilon_0^{-1}\mu\,\mathtt{nnz}(A) + \varepsilon_0^{-2} n k^{1+1/\mu}\log^2(k/\varepsilon_0))$ time, and the sketching dimension is $m_{H_S} = O(\varepsilon_0^{-2} k \log(k/\varepsilon_0))$. The space needed is $O(n + m_{H_S} d)$.

We can also use *length-squared sampling* to obtain subspace embeddings. In Section 7.2 we will give a data structure and algorithm that implement length-squared sampling.

To analyze length-squared sampling in the context of ridge regression, we need some observations about ridge regression.

Lemma 10 *Let A with $k = \mathit{rank}(A)$ have thin SVD $A = U\Sigma V^\top$, implying $\Sigma \in \mathbb{R}^{k\times k}$. For $\lambda > 0$, let $A_{(\lambda)} \overset{def}{=} \begin{bmatrix} A \\ \sqrt{\lambda} V V^\top \end{bmatrix}$. For $b \in \mathbb{R}^n$, let $\hat{b} \overset{def}{=} \begin{bmatrix} b \\ 0_d \end{bmatrix}$. Then for all $x \in \operatorname{im}(V)$,*

$$\|Ax - b\|^2 + \lambda\|x\|^2 = \|A_{(\lambda)}x - \hat{b}\|^2,$$

and

$$x^* \overset{def}{=} \operatorname{argmin}_{x\in\mathbb{R}^d} \|Ax - b\|^2 + \lambda\|x\|^2 = \operatorname{argmin}_{x\in\mathbb{R}^d} \|A_{(\lambda)}x - \hat{b}\|^2.$$

The matrix $A_{(\lambda)}$ has SVD $A_{(\lambda)} = \begin{bmatrix} U\Sigma D \\ \sqrt{\lambda} V D \end{bmatrix} D^{-1} V^\top$, where $D \overset{def}{=} (\Sigma^2 + \lambda I_k)^{-1/2}$, and $\|A_{(\lambda)}^+\|^2 = 1/(\lambda + 1/\|A^+\|^2)$. We have $\|A_{i,*}\|^2 \|A_{(\lambda)}^+\|^2 \geqslant \|U_{i,*}\Sigma D\|^2$ for $i \in [n]$.

Proof: Since $x \in \operatorname{im}(V)$ has $x = Vz$ for some $z \in \mathbb{R}^k$, and since $V^\top V = I_k$, it follows that $VV^\top x = Vz = x$, and so

$$\|A_{(\lambda)}x - \hat{b}\|^2 = \|Ax - b\|^2 + \|\sqrt{\lambda} V V^\top x - 0\|^2 = \|Ax - b\|^2 + \lambda\|x\|^2,$$

as claimed.

The SVD of $A_{(\lambda)}$ is $A_{(\lambda)} = \begin{bmatrix} U\Sigma D \\ \sqrt{\lambda} V D \end{bmatrix} D^{-1} V^\top$, where D is defined as in the lemma statement, since the equality holds, and both $\begin{bmatrix} U\Sigma D \\ \sqrt{\lambda} V D \end{bmatrix}$ and V have orthonormal columns, and D^{-1} has non-increasing nonnegative entries. Therefore $A_{(\lambda)}^+ = VD \begin{bmatrix} U\Sigma D \\ \sqrt{\lambda} V D \end{bmatrix}^\top$.

We have

$$A_{(\lambda)}^+ \hat{b} = VD^2\Sigma U^\top b = V\Sigma D^2 U^\top b, \tag{2}$$

using that Σ and D are diagonal matrices.

By the well-known expression $x^* = A^\top(AA^\top + \lambda I_n)^{-1} b$, and using the *not*-thin SVD $A = \hat{U}\hat{\Sigma}\hat{V}^\top$, with $\hat{\Sigma} \in \mathbb{R}^{n\times d}$ and $\hat{U}$ and $\hat{V}$ orthogonal matrices,

$$\begin{aligned} x^* &= \hat{V}\hat{\Sigma}\hat{U}^\top(\hat{U}\hat{\Sigma}\hat{\Sigma}^\top\hat{U}^\top + \lambda\hat{U}\hat{U}^\top)^{-1} b = \hat{V}\hat{\Sigma}\hat{U}^\top\hat{U}(\hat{\Sigma}\hat{\Sigma}^\top + \lambda I_n)^{-1}\hat{U}^\top b \\ &= \hat{V}\hat{\Sigma}(\hat{\Sigma}\hat{\Sigma}^\top + \lambda I_n)^{-1}\hat{U}^\top b = V\Sigma(\Sigma^2 + \lambda I_k)^{-1} U^\top b = V\Sigma D^2 U^\top b, \end{aligned} \tag{3}$$

where the next-to-last step uses that $\hat{\Sigma}$ is zero except for the top k diagonal entries of $\hat{\Sigma}$.

Comparing (2) and (3), we have $A_{(\lambda)}^+ \hat{b} = x^*$.

Using the expression for $A_{(\lambda)}^+$, $\|A_{(\lambda)}^+\|^2 = D_{1,1}^2 = 1/(\lambda + 1/\|A^+\|^2)$.

Finally, since $(A_{(\lambda)})_{i,*} = A_{i,*}$ for $i \in [n]$, and letting $\hat{U} = \left[\begin{smallmatrix} U\Sigma D \\ \sqrt{\lambda} VD \end{smallmatrix}\right]$,

$$\begin{aligned}\|A_{i,*}\|^2 \|A^+_{(\lambda)}\|^2 &= \|(A_{(\lambda)})_{i,*}\|^2 \|A^+_{(\lambda)}\|^2 \geqslant \|(A_{(\lambda)})_{i,*} A^+_{(\lambda)}\|^2 \\ &= \|\hat{U}_{i,*} D^{-1} V^\top V D \hat{U}^\top\|^2 = \|\hat{U}_{i,*}\|^2 = \|U_{i,*}\Sigma D\|^2,\end{aligned}$$

as claimed. ■

Definition 11 (ridge leverage-score sample, statistical dimension) *For $A, \lambda, A_{(\lambda)}$, and D as in Lemma 10, call $\mathcal{S} \subset [n]$ a* ridge leverage-score sample *of A if each $i \in \mathcal{S}$ is chosen independently with probability at least $\|U_{i,*}\Sigma D\|^2 / \mathit{sd}_\lambda(A)$, where the statistical dimension $\mathit{sd}_\lambda(A) \overset{def}{=} \|U\Sigma D\|_F^2 = \sum_i \sigma_i^2/(\lambda + \sigma_i^2)$, recalling that $U\Sigma D$ comprises the top n rows of the left singular matrix of $A_{(\lambda)}$.*

Definition 12 (length-squared sample) *Let $A \in \mathbb{R}^{n\times d}$, $\lambda > 0$, and $A_{(\lambda)}$ as in Lemma 10. For given $m_{\hat{L}}$, let matrix $\hat{L} \in \mathbb{R}^{m_{\hat{L}} \times n}$ be chosen by picking each row of $\hat{L}$ to be $e_i^\top / \sqrt{p_i m_{\hat{L}}}$, where $e_i \in \mathbb{R}^n$ is the i'th standard basis vector, and picking $i \in [n]$ with probability $p_i \leftarrow \|A_{i,*}\|^2 / \|A\|_F^2$.*

Lemma 13 (Length-squared sketch) *There is $m_{\hat{L}} = O(\nu \|A^+_{(\lambda)}\|^2 \|A\|_F^2 / \mathit{sd}_\lambda(A))$ so that a set of $m_{\hat{L}}$ length-squared samples contains a ridge leverage-score sample of A of size ν.*

Note that when $\lambda = 0$, $\|A^+_{(\lambda)}\| = \|A^+\|$, $\mathtt{sd}_0(A) = \mathtt{rank}(A)$, and the ridge leverage-score samples are leverage-score samples.

Proof: The expected number of times index $i \in [n]$ is chosen among $m_{\hat{L}}$ length-squared samples, $p_i m_{\hat{L}}$, is within a constant factor of

$$\frac{\|A_{i,*}\|^2}{\|A\|_F^2} \nu \|A^+_{(\lambda)}\|^2 \|A\|_F^2 / \mathtt{sd}_\lambda(A) \geqslant \frac{\|U_{i,*}\|^2}{\mathtt{sd}_\lambda(A)} \nu,$$

using Lemma 10, an expectation at least as large as for a ridge leverage-score sample of size ν. ■

Lemma 14 (Johnson-Lindenstraus Lemma) *For given $\varepsilon > 0$, if $P \subset \mathbb{R}^c$ is a set of $m \geqslant c$ vectors, and $G \in \mathbb{R}^{m_G \times c}$ has entries that are independent Gaussians with mean zero and variance $1/m_G$, then there is $m_G = O(\varepsilon^{-2} \log(m/\delta))$ such that with failure probability δ, $\|Gx\| = (1 \pm \varepsilon)\|x\|$ for all $x \in P$. Moreover, there is $m_G = O(\mu)$ so that $\|Gx\| \geqslant \|x\|/n^{1/\mu}$, with failure probability at most $1/n^2$.*

3 Sketching to Small Dimension using SparseGaussian

In this section, we show properties of the "GRHT"(Gaussian Randomized Hadamard Transform), which is the SRHT, mentioned in Lemma 9, multiplied by a SparseGaussian, defined just below. We show that this composition of sketching techniques gives a subspace ε-embedding to $O(k/\varepsilon^2)$ dimensions for a rank-k matrix. We also show that the SparseGaussian can be used for "projection-cost preserving" sketches, along the way showing that they support fast estimation of matrix products.

The key conceptual contribution of this section is that the simple GRHT construction removes a log factor from the size of canonical constructions of subspace embeddings and projection-cost preserving sketches, allowing various matrix computations to be done optimally, or nearly so.

Definition 15 (SparseGaussian) *A random matrix* $G \in \mathbb{R}^{m_G \times n}$ *is a* SparseGaussian *matrix if, for a given constant* c*, each of its entries is chosen independently, so that with probability* $\log^c(n)/n$*, it has distribution* $\mathcal{N}(0, 1/n)$*, and is zero otherwise.*

We begin by stating a composition theorem that obtains subspace embeddings for a combined SRHT and SparseGaussian sketch.

Theorem 16 (Composing SRHT and SparseGaussian) *Given* $\varepsilon > 0$ *and any* $n \times d$ *matrix* A*, let* H *be an SRHT sketch (see Lemma 9) with* $O(d\log(d)/\varepsilon^2)$ *rows and* G *be a SparseGaussian, with* $O(\frac{d}{\varepsilon^2})$ *rows. Then, with probability* $96/100$*, for all* $x \in \mathbb{R}^d$*,*

$$\|GHAx\|_2^2 = (1 \pm \varepsilon)\|Ax\|_2^2.$$

Further, GHA *can be computed in* $O(nd\log(nd))$ *time.*

As an immediate consequence of Theorem 16, we obtain the following very useful corollary, a "three-stage" sketching scheme that yields a subspace embedding computable in input-sparsity time:

Corollary 17 *Let* $A \in \mathbb{R}^{n\times d}$ *have* rank$(A) = k$*, and given* $\mu > 1$*. There is SparseGaussian* G_S *with* $m_{G_S} = O(k/\varepsilon_0^2)$ *rows, let* H_S *be a SRHT matrix and* T_S *be a ONSAP matrix such that* $H_S T_S$ *is a two-stage* ε_0*-embedding(as defined in Lemma 9), such that with constant failure probability,* $S = G_S H_S T_S$ *is an* ε_0*-embedding computable in* $O(\varepsilon_0^{-1}\mu\,\mathrm{nnz}(A) + \varepsilon_0^{-2} n k^{1+1/\mu}\log^2(k/\varepsilon_0))$ *time. The space needed is* $O(n + m_{G_S} d)$.

This corollary follows simply from observing that subspace embeddings are composable, T_S is a subspace embedding for A and Theorem 16 implies $G_S H_S$ is a subspace embedding for $T_S A$.

We now focus on proving Theorem 16. We begin by showing in Lemma 19 that the SRHT matrix applied to any matrix A flattens the vectors in the image (column span) of A. In other words, the image of HA comprises vectors with small ℓ_∞ norm, while preserving the Euclidean norms.

Then we show in Theorem 18 that under the assumption that all vectors in the image have bounded ℓ_∞ norm, multiplying on the left by a SparseGaussian sketch G does not dilate or contract any fixed vector by more than a $(1+\varepsilon)$-factor with high probability and thus yields a subspace embedding.

Theorem 18 (Subspace Embedding) *Let* A *be* $n \times d$ *matrix such that with probability* $99/100$*, for all* x*,* $\|Ax\|_\infty \leqslant \sqrt{\frac{\log(n)}{n}}\|Ax\|_2$*. Let* G *be a* $d/\varepsilon^2 \times n$ *SparseGaussian matrix. Then, with probability at least* $99/100$*, for all* x*,*

$$\|GAx\|_2^2 = (1 \pm \varepsilon)\|Ax\|_2^2,$$

that is, G *is a subspace* ε*-embedding for* A*.*

We prove the above theorem using the following lemmas on flattening, dilation and contraction of vector norms. As mentioned, the SRHT gives a good source of matrices with such "flat" column spans. We defer the proofs to Appendix B.

Lemma 19 (Flattening Transform.) *Given* $\delta > 0$*, a fixed vector* $x \in \mathbb{R}^d$*, and a* $n \times d$ *matrix* A*, let* H *be a* $\log(1/\delta) \times n$ *Subsampled Randomized Hadamard Transform matrix (as defined in Lemma 9). Then, with probability at least* $1 - \delta$*,* $\|HAx\|_\infty \leqslant c\|Ax\|_2\sqrt{\frac{\log(n/\delta)}{n}}$ *for some fixed constant* c*.*

Lemma 20 (Dilation.) *Given* $k \in [n]$ *and* $0 < \epsilon < 1$*, let* G *be a* $O(k/\varepsilon^2) \times n$ *SparseGaussian matrix. For a fixed vector* $y \in \mathbb{R}^n$ *such that* $\|y\|_\infty \leqslant \sqrt{\frac{c\log(n)}{n}}\|y\|_2$*, with probability at least* $1 - \exp(-k)$*,*

$$\|Gy\|_2^2 \leqslant (1+\varepsilon)\|y\|_2^2$$

Lemma 21 (Contraction.) *Given* $k \in [n]$ *and* $0 < \epsilon < 1$*, let* G *with a SparseGaussian matrix with* $O(k/\varepsilon^2)$ *rows such that each entry in* G *is* $\mathcal{N}(0, \varepsilon^2/k)$ *with probability* $\log^c(n)/n$*, for a fixed constant* c*. For a fixed vector* $y \in \mathbb{R}^n$ *such that* $\|y\|_\infty \leqslant \sqrt{\frac{c\log(n)}{n}}\|y\|_2$*, with probability at least* $1 - \exp(-k)$*,* $\|Gy\|_2^2 \geqslant (1-\varepsilon)\|y\|_2^2$*.*

Given the three lemmas above, we can prove Theorem 18 using standard arguments, as shown in Appendix B. We now have all the tools required to prove Theorem 16.

Proof: [of Theorem 16] It follows from Lemma 9 that with probability $99/100$, a SRHT sketch H with $\frac{d\log(d)}{\varepsilon^2}$ rows is a $(1 \pm \varepsilon)$-approximate subspace embedding. Let ζ_1 be the event this succeeds and condition on it. Then, from Theorem 18 it follows that with probability $99/100$ G with $k = O(d)$ is a $(1 \pm \varepsilon)$ subspace embedding for H. Let ζ_2 be the event this succeeds and condition on it. It then follows that for X,

$$\|GHAx\|_2^2 = (1 \pm \varepsilon)\|Hx\|_2^2 = (1 \pm \varepsilon)^2\|Ax\|_2^2 = (1 \pm 2\varepsilon)\|Ax\|_2^2$$

■

3.1 Fast Projection-Cost Preservation with the SparseGaussian

Here we show that the SparseGaussian can be used for *Projection Cost Preserving Sketches*; we will apply these to low-rank approximation. To do this, we show that SparseGaussian sketches satisfy the Approximate Matrix Product condition:

Definition 22 (Approximate Matrix Product (ε-AMM)) *Given matrices* A*,* B *and a sketching matrix* S*, we say* S *satisfies* ε*-Approximate Matrix Product for Frobenius norm if*

$$\|A^T S^T S B - A^T B\|_F \leqslant \varepsilon\|A\|_F\|B\|_F$$

similarly S *satisfies a stronger notion for operator norm if*

$$\|A^T S^T S B - A^T B\|_2 \leqslant \varepsilon\|A\|_2\|B\|_2$$

We use the following simple lemma for the composability of sketches that satisfy Approximate Matrix Product:

Lemma 23 (Composability of Approximate Matrix Product) *Given $\varepsilon > 0$ and sketching matrix S_1 and S_2 such that both satisfy ε-Approximate Matrix Product, and preserve Frobenius norm up to a constant, it follows that*

$$\|A^T S_1^T S_2^T S_2 S_1 B - A^T B\|_F \leqslant O(\varepsilon)\|A\|_F \|B\|_F$$

Lemma 24 (Approximate Matrix Product for Standard Sketches, [CNW15, Coh16]) *For any $\varepsilon > 0$, OSNAP and SRHT matrices with $O(d/\varepsilon^2)$ rows satisfy the ε-Approximate Matrix Product property from definition 22, with probability at least $99/100$.*

Lemma 25 (Approximate Matrix Product for SparseGaussian) *Given matrices $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times d}$, let G be a Sparse Gaussian matrix with d/ε^2 rows and $\mathrm{poly}\log(n)$ non-zeros. Then, with probability $99/100$,*

$$\|A^T G^T G B - A^T B\|_F \leqslant \varepsilon\|A\|_F\|B\|_F$$

Definition 26 (Projection Cost Preserving Sketch (PCP) [CEM⁺15]) *Given $A \in \mathbb{R}^{n'\times d}$, $\tilde{A}$ is a rank-k Projection Cost Preserving sketch for A with error $\varepsilon \geqslant 0$, in short, a (k, ε)-PCP, if for all rank-k projection matrices P,*

$$(1-\varepsilon)\|A - AP\|_F^2 \leqslant \|\tilde{A} - \tilde{A}P\|_F^2 + C \leqslant (1+\varepsilon)\|A - AP\|_F^2$$

An OSNAP, SRHT and SparseGaussian can be composed to obtain a PCP for A.

Lemma 27 (Composing Sketches to get a PCP) *Let $A \in \mathbb{R}^{n\times d}$. Let C be a OSNAP matrix with $O(k^{1.01})$ rows. Let PHD be an SRHT with $O(\frac{k\log(k)}{\varepsilon})$ rows and let G be a SparseGaussian with $O(\frac{k}{\varepsilon})$ rows. Compute $GPHDCA$.*

Then, with probability $99/100$, $\tilde{A} = GPHDC$ is a rank-k projection cost preserving sketch for A.

4 Rank computation and Independent Row Selection

In this section we sharpen the results of [CKL13], clarifying the space needed and removing some logarithmic factors in running times. We confine our attention to the field of real numbers, and assume unit cost operations there. The results of [CKL13] apply in this setting; we ignore questions of the bit-complexity of arithmetic operations.

4.1 Rank Computation

We will need *rank-preserving sketching matrices*, defined as follows.

Definition 28 (Rank-preserving sketches) *A random distribution over matrices $S \in \mathbb{R}^{z_S\times n}$ is* rank-preserving *if there is a constant $c > 0$ such that with high probability, for a given matrix A, a matrix S with that distribution has $\mathit{rank}(SA) \geqslant \min\{z_S/c, \mathit{rank}(A)\}$.*

There are simple and sparse constructions of rank-preserving sketching matrices.

Theorem 29 (Rank-preserving sketch construction [CKL13]) *There are rank-preserving sketching distributions as above so that:*

- *SA can be computed in $O(\mathtt{nnz}(A))$ time;*
- *S has two non-zero entries per column;*
- *S has at most $2n/z_S + 2$ nonzero entries per row.*

(It is also true and unsurprising that count-sketch matrices have qualitatively similar properties.)

Due to their sparsity and its regularity, rank-preserving sketches SA can be quickly updated when single entries of A are changed.

Lemma 30 *For $A \in \mathbb{R}^{n \times d}$, and rank-preserving sketches $S \in \mathbb{R}^{z_S \times n}$ and $R^\top \in \mathbb{R}^{z_R \times d}$, SAR can be updated in $O(1)$ time when a single entry of A is changed (or $O(\mathtt{nnz}(A))$ if A is given as a whole), and using $O(\min\{\mathtt{nnz}(A), z_S z_R\})$ space.*

Proof: By hypothesis S and $R^\top$ have $O(1)$ nonzero entries per column. If A has a single nonzero entry a_{ij}, then $SAR = a_{ij} S_{*,i} R_{j,*}$, which has $O(1)$ nonzero entries altogether. So updating for a given entry takes $O(1)$ time, since those $O(1)$ nonzero entries can be determined and accessed in $O(1)$ time. (For example, we store each row of SAR in a hash table.) By this means, we compute SAR without computing SA as an intermediate step, and so need at most the space $O(\min\{\mathtt{nnz}(A), z_S z_R\})$ needed to store $SAR \in \mathbb{R}^{z_S \times z_R}$. ■

Remark 31 We interchangeably refer to the aforementioned rank-preserving sketch as a MagicGraph sketch.

Theorem 32 (Rank computation) *Let $k = \mathsf{rank}(A)$. Given value $s \geqslant \sqrt{n}$, the quantity $k_0 \overset{def}{=} \min\{k, s\}$ can be determined by Algorithm 1 (*RankEstBounded*) with failure probability $O(1/d^{1/3})$, using $O(s^2)$ space and $O(\mathtt{nnz}(A) + \min\{k_0^\omega, k_0 \,\mathtt{nnz}(A)\})$ time. Moreover, $\mathsf{rank}(A)$ can be determined by Algorithm 2 (*RankEst*) in $O(\mu\, \mathtt{nnz}(A) + \min\{k^\omega, k\,\mathtt{nnz}(A)\})$ time and $O(n + k^{2+1/\mu})$ space, for given $\mu \geqslant 1$.*

Algorithm 1 RankEstBounded(A, s)

Input: $A \in \mathbb{R}^{n \times d}$, integer $s > 0$

Output: $k_0 \overset{def}{=} \min\{s, \mathsf{rank}(A)\}$

```
// Using CKL-RE, the algorithm of Theorem 2.6 of [CKL13]
$z \leftarrow c * \min\{s, (\mathtt{nnz}(A)/\log n)^{1/2}\}$     // $c \geqslant 1$ is a constant
Generate rank-preserving sketches $S_1 \in \mathbb{R}^{z \times n}$ and $R_1^\top \in \mathbb{R}^{z \times d}$
Compute $S_1 A R_1$     // using Lemma 30
$k_1 \leftarrow \mathsf{rank}(S_1 A R_1)$     // using CKL-RE
if $k_1 < z/c$ or ($k_1 == z/c$ and $z/c == s$):
    return $k_1$
Generate rank-preserving sketches $S_2 \in \mathbb{R}^{cs \times n}$ and $R_2^\top \in \mathbb{R}^{cs \times d}$
Compute $S_2 A R_2$     // using Lemma 30
$k_2 \leftarrow \mathsf{rank}(S_2 A R_2)$     // using CKL-RE
return $\min\{s, k_2\}$
```

Algorithm 2 $\textsc{RankEst}(A, \mu)$

```
Input: $A \in \mathbb{R}^{n \times d}$, $\mu \geqslant 1$
Output: $k = \mathrm{rank}(A)\}$
for $t = 0, 1, \ldots$:
    $s_t = n^{1/2 + t/\mu}$
    $k_0 \leftarrow \textsc{RankEstBounded}(A, s_t)$
    if $k_0 < s_t$:
        return $k_0$
```

Proof: The CKL-RE algorithm includes the computation and use of two applications of rank-preserving sketches such as S_1AR_1, and the algorithm failure probability is $O(1/d^{1/3]})$, so with that failure probability, we assume the correctness conditions for CKL-RE and for the sketches S_iAR_i here. That is, with failure probability $O(1/d^{1/3})$, $k_i = \mathrm{rank}(S_iAR_i)$, for $i \in \{1, 2\}$, and $\min\{z/c, \mathrm{rank}(S_1AR_1)\} = \min\{z/c, \mathrm{rank}(A)\}$, and also $\min\{s, \mathrm{rank}(S_2AR_2)\} = \min\{s, \mathrm{rank}(A)\}$.

Under these assumptions, Algorithm 2 returns $\min\{s, \mathrm{rank}(A)\}$: since $k_1 = \min\{z/c, \mathrm{rank}(A)\}$, if $k_1 < z/c \leqslant s$, the $k_1 = \mathrm{rank}(A) < s$, and k_1 as returned in Step 6 is equal to k_0. Also, if $k_1 = z/c = s$, then $s \leqslant \mathrm{rank}(A)$, and then also the returned k_1 is equal to k_0. If $\min\{s, k_2\}$ is returned, then since that quantity is equal to $\min\{s, \mathrm{rank}(A)\} = k_0$, the algorithm is also correct in that case.

The running time to compute S_1AR_1 is $O(\min\{\mathrm{nnz}(A), z^2\}) = O(\mathrm{nnz}(A)/\log d)$, by Lemma 30, and to compute $\mathrm{rank}(S_1AR_1)$ is

$$O(\mathrm{nnz}(S_1AR_1) \log k_1 + \min\{k_1^\omega, k_1 \,\mathrm{nnz}(S_1AR_1)\}),$$

by Theorem 2.6 of [CKL13]. Since $\mathrm{nnz}(S_1AR_1) \log k_1 = O(z^2 \log d) = O(\mathrm{nnz}(A))$, and

$$k_1 \leqslant \min\{z, \mathrm{rank}(A)\} \leqslant \min\{cs, \mathrm{rank}(A)\} \leqslant ck_0,$$

the running time up to Step 6 is within the claimed bounds.

If the algorithm continues to the computation of k_2, it must hold that $k_1 \geqslant z/c$ and $z/c > s$, so that $z/c = (\mathrm{nnz}(A)/\log n)^{1/2}$, and it follows that both $\mathrm{rank}(A)$ and s are greater than $(\mathrm{nnz}(A)/\log n)^{1/2}$, that is, $k_0 \geqslant (\mathrm{nnz}(A)/\log n)^{1/2}$. We have, using $\mathrm{nnz}(A) \geqslant n$ and $\omega > 2$, and assuming n large enough,

$$k_0^\omega \geqslant (\mathrm{nnz}(A)/\log n)^{\omega/2} \geqslant \mathrm{nnz}(A) \log n.$$

Therefore, using this condition and again Theorem 2.6 of [CKL13], the computation of $\mathrm{rank}(S_2AR_2)$ takes

$$O(\mathrm{nnz}(A) \log k_2 + \min\{k_2^\omega, k_2 \,\mathrm{nnz}(S_2AR_2)\}) = O(\min\{k_0^\omega, k_0 \,\mathrm{nnz}(A)\}),$$

using also $k_2 \leqslant k_0$. That is, the overall time needed is $O(\mathrm{nnz}(A) + \min\{k_0^\omega, k_0 \,\mathrm{nnz}(A)\})$, as claimed.

The space needed to compute and store the sketches S_iAR_i is $O(\min\{\mathrm{nnz}(A), s^2\})$, via Lemma 30, and the space needed by algorithm CKL-RE is also $O(s^2)$, to do rank computations on matrices that are in $\mathbb{R}^{O(s) \times O(s)}$.

So under all relations between $\mathrm{rank}(A)$, s, and $\mathrm{nnz}(A)$, the space used is bounded by $O(s^2)$, and the running time is at most $O(\mathrm{nnz}(A) + \{k_0^\omega, k_0 \,\mathrm{nnz}(A)\})$. All but the last statement of the theorem follow.

Algorithm 2 (RankEst) satisfies the last conditions of the theorem. If that algorithm exits for $t = 0$, the space used is $O(n)$. Otherwise, $k \geqslant k_0 \geqslant n^{1/2}$ If $k_0 = s_{t-1}$ for $t > 0$, then $s_{t-1} \leqslant k$, and so $s_t^2 \leqslant s_{t-1}^2 n^{2/\mu} \leqslant k^{2+4/\mu}$, using $k \geqslant n^{1/2}$. So $O(n + k^{2+4/\mu})$ space suffices, and the number of trials is at most μ. Adjusting constants, and noting that the terms $\min\{k_0^\omega, k_0\,\mathtt{nnz}(A)\}$ are dominated those terms for the last k_0 computed, the theorem follows. ■

4.2 Independent Row Selection

We can similarly sharpen an algorithm for finding k linearly independent rows/columns of a rank-k matrix. The main theorem we obtain in this section is as follows:

Theorem 33 (Finding a Row Basis via Selecting Independent Rows) *Given $n \times d$ matrix A such that $k = \mathit{rank}(A)$, there exists an algorithm that with probability $99/100$ outputs a set of k independent rows of A and runs in $O(\mathtt{nnz}(A) + k^\omega \log\log(n))$ time, where ω is the matrix multiplication constant.*

We first recall the algorithm of Cheung et. al. to find a subset of r independent rows of a rank r, matrix A when r is known:

Lemma 34 (Linearly Independent Rows [CKL13]) *Given a rank r, $n \times d$ matrix A, there exists an algorithm that with probability at least $99/100$, outputs a set of r linearly independent rows of A and runs in time $O(\mathtt{nnz}(A) + r^\omega \log(n/r))$.*

We first observe that in the proof of the above Lemma (Theorem 2.7 of [CKL13]) their algorithm, needing k as input, runs in $O((\mathtt{nnz}(A) + k^\omega) \log n)$ time; examination of their proof shows that the $\log n$ term can be replaced by $\log(n/k)$. It also needs $O(kd)$ space, as can be readily derived. We will call this algorithm `CKL-LI`.

Now, we are ready to describe our algorithm. We begin with a MagicGraph sketch matrix T that we use to decrease the number of columns of A. Since T is a MagicGraph sketch with r columns, it follows from Lemma 29 that $\mathsf{rank}(A) = \mathsf{rank}(AT)$. Further, $\mathtt{nnz}(AT) = O(\mathtt{nnz}(A))$ and if a set of r rows of A are independent, the corresponding set of r rows in AT are also independent. Therefore, it suffices to focus on finding a set of r independent rows for AT.

Next, we show that we can preprocess the matrix AT such that the number of non-zeros can by reduced by a $\operatorname{poly}\log(n)$ factor and the preprocessing preserves independent rows in A.

Lemma 35 (Pre-processing) *Given a $n \times d$ rank r matrix A such that $\mathtt{nnz}(A) = \Omega(r^2)$, with probability at least $99/100$, Algorithm 4 outputs a $n' \times d$ matrix A' such that $n' \leqslant n$ and $\mathtt{nnz}(A') \leqslant \mathtt{nnz}(A)/\log^c(n)$, for a fixed constant c, $\mathit{rank}(A') = \mathit{rank}(A)$ and a set of independent rows in A are independent in A'. Further, Algorithm 4 runs in $O(\mathtt{nnz}(A) + r^\omega \log\log(n))$ time.*

Proof: Here we assume $\mathtt{nnz}(A) \geqslant r\mathrm{polylog}(r)$, else the running time is dominated by r^ω. To see this, consider the truncated identity matrix as the input. We cannot decrease the sparsity without decreasing the rank. But this assumption will hold in our setting since if $\mathtt{nnz}(A) < r^2$, the running time would be dominated by r^ω. Consider the first iteration of Algorithm 4. Let κ be a fixed large constant. Since we randomly hash n rows of AT to $2^\kappa \cdot r$ buckets each bucket contains $\Theta(2^\kappa)$ rows with probability at least $1 - 1/\mathrm{poly}(n)$, using standard balls in bins arguments. Since we then compute a set of r independent rows of the resulting matrix and consider the indices of AT

Algorithm 3 FindBasis($A, \texttt{dim} = \texttt{row}$)

Input: $A \in \mathbb{R}^{n\times d}$, `dim` flags for row or column basis, default is row basis

Output: k, Λ, where $k = \mathsf{rank}(A)$, and $\Lambda \in \mathbb{R}^{k\times n}$ has rows a subset of the rows of the identity matrix, selecting a subset $\mathcal{S} \subset [n]$ of k linearly independent rows of A.

```
if dim==col
    $A \leftarrow A^\top$
Run Algorithm 2 on $A$ to compute $r = k = \mathsf{rank}(A)$.
Let $T$ be a $d \times r$ MagicGraph matrix as defined above. Let the resulting matrix be $AT$. Now, we have reduced the problem to finding a subset of $n$ rows of $AT$ with full rank and the time was $O(\mathtt{nnz}(A) + r^\omega)$.
Pre-process $AT$ such that $\mathtt{nnz}(A)/\text{poly}\log(n)$ by running Algorithm 4. Let $W$ be a $r\log(r) \times n$ OSNAP matrix (as defined in 9) and let $WAT$ be a subspace embedding for $AT$. Let $PHD$ be a SRHT such that $P$ has $O(r)$ rows. Compute $\widetilde{A} = PHDWAT$.
Let $G$ be a $r \times r\log(r)$ SparseGaussian matrix such that $G_{i,j} = \mathcal{N}(0, 1/r)$ with probability $\log^c(r)/r$, for a fixed constant $c$, and $0$ otherwise. Compute $G\widetilde{A}$ and let $QR^{-1}$ be the QR factorization for $G\widetilde{A}$. Let $G'$ be a dense $r \times \log(n)$ Gaussian matrix.
For all $i \in [n]$, let $\hat{\tau}_i = \|e_i^T ATRG'\|_2^2$ be the approximation to the $i$-th leverage score of $AT$. Let $p = \{p_1, p_2, \ldots p_n\}$ be a probability distribution over the rows of $AT$ such that $p_i = \hat{\tau}_i / \sum_{i\in[n]} \hat{\tau}_i$. Sample $r\log(r)$ rows of $AT$ proportional to $p$ and let the resulting matrix be $SAT$.
Let $\mathcal{S}_0 = [r\log(r)]$ and let $c'$ be a fixed constant. For $\ell \in [O(\log(\log(n))]$,
    - Let $M_\ell$ be a $(2 \uparrow\uparrow \ell)^{c'} r \times r\log(r)$ MagicGraph. Compute $M_\ell(SAT)_{|\mathcal{S}_{\ell-1}}$.
    - Compute a set of independent rows of $M_\ell(SAT)_{|\mathcal{S}_{\ell-1}}$ by running the Algorithm CKL-LI and let $\mathcal{S}_\ell$ be the resulting indices in $SAT$ that map to a set of $r$ linearly independent rows.
The resulting matrix $(SAT)_{|\mathcal{S}_{c'\log\log(n)}}$ only has $100r$ rows and is of full rank. Run Algorithm CKL-LI on this matrix and compute a set of independent rows. Let $\mathcal{S}$ be the preimage of these rows in in $AT$.
Construct row selection matrix $\Lambda$ from $\mathcal{S}$
return $k, \Lambda$
```

Algorithm 4 Preprocessing to Decrease $\texttt{nnz}(A)$

```
Input: A $n \times d$ matrix $A$, $r = \mathsf{rank}(A)$, and a constant $c' \in \mathcal{N}$.
Output: Matrix $A_{|\mathcal{T}_{c\ln\ln(n)}} \in \mathbb{R}^{n' \times d}$ with $\texttt{nnz}(A_{|\mathcal{T}_{c\ln\ln(n)}}) \leqslant \frac{\texttt{nnz}(A)}{\log^{c'}(n)}$ and $\mathsf{rank}(A_{|\mathcal{T}_{c\ln\ln(n)}}) = \mathsf{rank}(A)$

$\mathcal{T}_0 \leftarrow [n]$, let $c$ be a fixed constant
for $\ell = 1, 2, \dots c \log\log(n)$:
        $A_\ell \leftarrow A_{|\mathcal{T}_{\ell-1}}$
        Compute $MA_\ell$, where $M$ is a $100r \times n$ $\mathsf{MagicGraph}$ matrix
                // $\texttt{nnz}(MA_\ell) = O(\texttt{nnz}(A_\ell))$ since $M$ has $O(1)$ non-zeros in each column.
        Compute $\mathcal{S}_\ell$, a subset of $r$ independent rows of $MA_\ell$
        Let $\mathcal{T}_\ell$ be the indices of rows of $A_\ell$ that mapped to $\mathcal{S}_\ell$.
return $A_{|\mathcal{T}_{c\ln\ln(n)}}$
```

that hash to this set, we obtain a subset of $\Theta(r2^\kappa)$ rows indexed by the set $\mathcal{T}_1$. It follows from $\texttt{CKL-LI}$ that finding a set of r linearly independent rows requires $O(\texttt{nnz}(AT) + r^\omega \log(r \cdot 2^\kappa/r)) = O(\texttt{nnz}(A) + \kappa \cdot r^\omega)$ time.

We show that $(AT)_{\mathcal{T}_1}$ has at most $O(\texttt{nnz}(AT)/2^\kappa + r^2)$ non-zeros with high probability. Recall, since $\mathsf{MagicGraph}$ preserves the rank of a matrix with probability $99/100$, there exists a subset of some r independent rows in AT which hash to distinct buckets. Let ζ_1 be the event that the rank is preserved, and let $\mathcal{I}$ denote the index of r rows that hash to distinct buckets. Observe, we do not decrease the sparsity in these rows at all and in the worst case, we are can upper bound the total sparsity of rows in $\mathcal{I}$ by $O(r^2)$ For the remaining rows, each one is equally likely to be hashed to one of the $2^\kappa r$ independent buckets. For all $i \in [n] \setminus \mathcal{I}$, let X_i^1 denote the indicator for the first bucket. It is easy to see that $\mathbb{E}\left[\sum_{i \in [n] \setminus \mathcal{I}} \mathcal{X}_i^1\right] = n/2^\kappa r$. Further, by Chernoff,

$$\Pr\left[\sum_{i \in [n] \setminus \mathcal{I}} X_i^1 > \frac{\log(n)}{2^\kappa r} n\right] \leqslant \exp\left(-\frac{\varepsilon^2 \log(n) n}{2^\kappa r}\right)$$

Since $2^\kappa \cdot r \leqslant n$, it follows that each bucket has at most $O(n\log(n)/2^\kappa r)$ rows with $1 - 1/\mathrm{poly}(n)$ probability. Since the number of buckets is at most n, union bounding over the failure probability of each such Chernoff bound we can conclude that with probability at least $1 - 1/\mathrm{poly}(n)$, for all $j \in [2^\kappa r]$, $\sum_{i \in [n] \setminus \mathcal{I}} X_i^j = O(n\log(n)/2^\kappa r)$. Since we select a set of r independent rows, we can conclude that $\texttt{nnz}((AT)_{\mathcal{T}_1}) = O(\texttt{nnz}(AT)\log(n)/2^\kappa + r^2)$.

Subsequently, we recurse on the subset of rows indexed by $\mathcal{T}_1$. For each $\ell \in [c\log\log(n)]$, we create a fresh $\mathsf{MagicGraph}$ matrix and hash to $r \cdot (2 \uparrow\uparrow \ell)^\kappa$ buckets. We then find a set of r independent rows in the resulting matrix. The running time for finding a subset of r independent rows is therefore $\texttt{nnz}((AT)_{\mathcal{T}_{\ell-1}}) + \kappa \cdot (2 \uparrow\uparrow \ell - 1) r^\omega$. The subset of rows of AT left after the ℓ-th iteration are $\frac{n}{(2\uparrow\uparrow\ell-1)}$. Analysing the above recurrence we observe that the running time dependence on the sparsity is a geometric series and the dependence on the rank has the form $k^\omega \log\left(\prod_{\ell \in} |\mathcal{T}_\ell|\right)$. Combining the above observations, we note that $\log\log(n)$ recursion steps suffice to decrease sparsity to $\texttt{nnz}(AT)/\log^c(n)$. Note, at each step the success probability is at least $1 - 1/\mathrm{poly}(n)$ and thus union bounding over the failure probability of each step results in an algorithm that succeeds with probability at least $99/100$. Since the running time is a geometric series, we can set

κ to be an appropriately large constant and the claim follows. ■

Since W is a OSNAP with $r\log(r)$ rows, WAT is an $O(1)$-approximation subspace embedding for the column span of AT. Since $\mathtt{nnz}(AT) = \mathtt{nnz}(A)/\log^c(n)$, WAT can be computed in $O(\mathtt{nnz}(A))$ time.

We will need the following standard fact for computing leverage scores; since our algorithm for leverage scores begins with finding a basis, we will use older algorithms to avoid circularity.

Lemma 36 (Naive Leverage Score Computation) *Given a κ-approximate oblivious subspace embedding S for an $n \times d$ matrix A, there exists an algorithm to compute $O(\kappa)$-approximate leverage scores for A in $O((\mathtt{nnz}(A) + d^{\omega})\log(n))$ time.*

Proof: This is standard and follows from computing a QR decomposition for SA such that $QR^{-1} = SA$. Let G be a $r \times \log(n)$ dense Gaussian matrix. Then for all $i \in [n]$, the row leverage scores of A are given by $\tau_i = \|e_i^T ARG\|_2^2$. ■

Next, we prove the Algorithm 3 indeed outputs a set of r independent rows in $\mathtt{nnz}(A) + r^{\omega}\log\log(n)$ time.

Proof: [Proof of Theorem 33] Since the SRHT is a $O(1)$-approximate oblivious subspace embedding for WAT, it follows from Lemma 19, that with probability at least 99/100, for all $x \in \mathbb{R}^d$,

$$\frac{1}{O(1)}\|WATx\|_2^2 \leqslant \|PHDWATx\|_2^2 \leqslant \|WATx\|_2^2$$

and

$$\|PHDWATx\|_\infty \leqslant c\sqrt{\frac{\log(k/\delta)}{k}}\|WATx\|_2$$

Let ζ_1 be the event that the two aforementioned conditions hold. Next, observe that the matrix $\widetilde{A} = PHDWAT$ satisfies all the preconditions for Theorem 18. Given a SparseGaussian sketch G with $O(r)$ rows, it follows that with probability at least 99/100, for all $x \in \mathbb{R}^r$,

$$\frac{1}{2}\|\widetilde{A}x\|_2^2 \leqslant \|G\widetilde{A}x\|_2^2 \leqslant 2\|\widetilde{A}x\|_2^2 \tag{4}$$

Let ζ_2 be the event that the above equation is satisfied by G. Therefore, conditioned on ζ_2, G is a $\Theta(1)$-approximate subspace embedding for $\widetilde{A}$. Let $QR^{-1} = G\widetilde{A}$ be the QR Decomposition for $G\widetilde{A}$. Let G' be a dense Gaussian matrix. By Lemma 36, it follows that with probability 99/100, for all $i \in [n]$ $\hat{\tau}_i = \|e_i ATRG'\|_2^2$ are $O(1)$-approximate leverage scores for AT. Let ζ_3 be the event that this holds. Then, conditioned on ζ_3, for all $i \in [n]$,

$$\frac{1}{O(1)}\tau_i(AT) \leqslant \hat{\tau}_i \leqslant \tau_i(AT)$$

Let $p = \{p_1, p_2, \dots p_n\}$, be a distribution over the rows of A such that $p_i = \hat{\tau}_i / \sum_{i\in[n]} \hat{\tau}_i$. Therefore, sampling $r' = r\log(r)$ rows of AT proportional to p and rescaling the rows results in a subspace embedding. Formally, let SAT be a random matrix such that for all $i \in [r']$, $(SAT)_{i,*} = \frac{1}{\sqrt{r'p_j}}(AT)_{j,*}$ with probability p_j. Then, for all $x \in \mathbb{R}^r$, with probability at least 99/100,

$$\frac{1}{O(1)}\|ATx\|_2^2 \leqslant \|SATx\|_2^2 \leqslant \|ATx\|_2^2$$

and let ζ_4 be the corresponding event.

Here, we observe that SAT is a scaled subset of the rows of AT and conditioned on ζ_4, $\mathsf{rank}(SAT) = \mathsf{rank}(AT)$, since the norms of all vectors are preserved up to a constant. Therefore, it suffices to find a subset of r independent rows of SAT. However, recall, SAT has $r\log(r)$ rows and naively computing a set of r independent rows incurs a logarithmic dependence on r which we set out to avoid. Instead, we run an iterative algorithm that sketches on the left with a $\mathsf{MagicGraph}$ matrix and computes a set of r independent rows of the resulting matrix. The algorithm and analysis follows that of the preproccessing step (Lemma 35) closely as we find r independent rows in the sketched matrix, restrict to the preimage of these rows in the original matrix and recurse. We repeat this $O(\log\log(n))$ times to decrease the the number of rows from $O(r\log(r))$ to $O(r)$, at which point running the Algorithm $\texttt{CKL-LI}$ suffices.

Recall, $\mathcal{S}_0 = [r\log(r)]$ and SAT is a $r\log(r) \times r$ matrix. For $\ell = 1$, recall, M_1 is a $\mathsf{MagicGraph}$with $2^\kappa r$ rows and we compute $M_1 SAT$. It follows that with probability $99/100$, $\mathsf{rank}(M_1 SAT) = \mathsf{rank}(SAT) = r$. We denote this event by ζ_5 and condition on it. Then, we compute a subset of r independent rows of $M_1 SAT$ which requires time $\mathtt{nnz}(SAT) + \kappa r^\omega$. Further, the preimage of these rows is at most $O(r\log(r)/\kappa)$ rows in SAT. To see this recall, each row of SAT is equally likely to be hashed to any one of the $2^\kappa r$ buckets in M_1. Then, for all $i \in [r\log(r)]$, let X_i^j denote the indicator for the i-th row of SAT hashing to the j-th bucket. It is easy to see that for a fixed j, $\mathbb{E}\left[\sum_{i\in[r\log(r)]} \mathcal{X}_i^j\right] = r\log(r)/2^\kappa r$. Further, by Chernoff,

$$\Pr\left[\sum_{i\in[r\log(r)]} X_i^j > \frac{\log(r)}{2^\kappa r} r\log(r)\right] \leqslant \exp\left(-\frac{\varepsilon^2 \log(r) r\log(r)}{2^\kappa r}\right)$$

Since $2^\kappa \cdot r \leqslant r\log(r)$, it follows that each bucket has at most $O(r\log(r)/2^\kappa r)$ rows with $1 - 1/\mathrm{poly}(r)$ probability. Since the number of buckets is at most $r\log(r)$, union bounding over the failure probability of each such Chernoff bound we can conclude that with probability at least $1 - 1/\mathrm{poly}(n)$, for all $j \in [2^\kappa r]$, $\sum_{i\in[r\log(r)]} X_i^j = O(r\log(r)/2^\kappa r)$. Recall, these rows are indexed by $\mathcal{S}_1$ and the sparsity of SAT restricted to $\mathcal{S}_1$ is a $1/2^\kappa$ factor smaller (this analysis appears in Lemma 35). Using similar arguments as before, we know that the set of independent rows can be computed in $O(\kappa r^\omega)$ time.

Subsequently, we recurse on the subset of rows indexed by $\mathcal{S}_1$. For each $\ell \in [c\log\log(n)]$, we create a fresh $\mathsf{MagicGraph}$ matrix and hash to $r \cdot (2 \uparrow\uparrow \ell)^\kappa$ buckets. We then find a set of r independent rows in the resulting matrix. The running time for finding a subset of r independent rows is therefore $\mathtt{nnz}((SAT)_{\mathcal{T}_{\ell-1}}) + \kappa \cdot (2 \uparrow\uparrow \ell - 1) r^\omega$. The subset of rows of SAT left after the ℓ-th iteration are $\frac{n}{(2\uparrow\uparrow\ell-1)}$. Observe, we obtain the resulting recurrence we obtain is similar to the one appearing in Lemma 35. Therefore, recursing $\log\log(n)$ times suffices to decrease sparsity to $\mathtt{nnz}(SAT)/\log^c(n)$. Further, observe the number of rows remaining in $(SAT)_{|\mathcal{S}_{\log\log(n)}}$ is at most $O(r)$. Note, at each step the success probability is at least $1 - 1/\mathrm{poly}(r)$ and thus union bounding over the failure probability of each step results in an algorithm that succeeds with probability at least $99/100$. Since the running time is a geometric series, we can set κ to be an appropriately large constant.

Since the final matrix is $O(r)$ rows and columns, we can afford to run the algorithm from Cheung et. al. [CKL13] to compute a set of r independent rows and it is easy to see that we can then back out a set r independent rows of A from this set. The running time of the iterative process is the same as the running time of the preprocessing step in Lemma 35. It follows from Lemma 34, that with probability 99/100, computing a set of r linearly independent rows of $(SAT)_{|\mathcal{S}_{c'\log\log(n)}}$ requires $O(r^2 + \log(cr/r)r^{\omega}) = O(r^{\omega})$ time. Denoting the success of the final event by ζ_6 and union bounding over all previous events, it follows that Algorithm 3 succeeds with 9/10 probability. This completes the proof of Theorem 33. ■

4.3 Applications

In this section, we describe corollaries that can be derived from our new sampling and sketching ideas, which are also used in the remainder of the paper.

4.4 Low Rank Approximation

```
Algorithm 5 Low-Rank Approximation
Input: $n \times d$ matrix A, $k \in [d]$, $\varepsilon > 0$

Let $C$ be a OSNAP matrix with $O(k^{1.01})$ rows. Let $PHD$ be a SRHT with $O(\frac{k\log(k)}{\varepsilon^2})$ rows and let $G$ be a SparseGaussian with $O(\frac{k}{\varepsilon})$ rows. Compute $GPHDCA$.
Compute the SVD for $GPHDCA$, denoted by $P\Lambda Q^T$.
Output: $QQ^T$
```

In this subsection we provide an optimal algorithm for Low-Rank Approximation under the current fast matrix multiplication time. The main theorem we prove is as follows:

Theorem 37 (LRA in Current Matrix Multiplication Time) *Given* $\varepsilon > 0$, *a* $n \times d$ *matrix* A *and* $k \in [n]$, *Algorithm 5 outputs a* $d \times k$ *matrix* Q *with orthonormal columns such that with probability* 9/10,

$$\|A - AQQ^T\|_F^2 \leqslant (1+\varepsilon)\|A - A_k\|_F^2$$

Further, Algorithm 5 runs in $O\left(\mathrm{nnz}(A) + \frac{dk^{\omega-1}}{\varepsilon} + \frac{dk^{1.01}\log^c(k)}{\varepsilon^2}\right)$ *time.*

Proof: [Theorem 37] By Lemma 27, with probability 99/100, $\tilde{A} = GPHDCA$ is a projection=cost preserving sketch for A. This implies, for all rank-k projection matrices P,

$$(1-\varepsilon)\|A - AP\|_F^2 \leqslant \|\tilde{A} - \tilde{A}P\|_F^2 + c \leqslant (a+\varepsilon)\|A - AP\|_F^2 \tag{5}$$

Since we compute the SVD of $\tilde{A}$, we obtain a rank-k projection matrix QQ^T such that

$$\|\tilde{A} - \tilde{A}QQ^T\|_F^2 = \|\tilde{A} - \tilde{A}_k\|_F^2$$

Therefore using equation 5 we can conclude that

$$\|A - AQQ^T\|_F^2 \leqslant \frac{(1+\varepsilon)}{(1-\varepsilon)}\|A - A_k\|_F^2 \leqslant (1+2\varepsilon)\|A - A_k\|_F^2$$

■

4.5 Graphs and Matroids.

Next, we discuss how our algorithm can be used to compute the Maximum Matching in an undirected Graph. Maximum Matching is a central problem in graph theory and has been extensively studied in various settings.

Corollary 38 (Maximum Matching) *Given an undirected, unweighted graph* $G(V, E)$ *and* $k \in [n]$, *there exists an algorithm that with probability* $99/100$ *outputs a matching in* G *of size* $\min(k, \mathit{opt})$, *where* opt *is the size of the maximum matching. Further, the algorithm runs in time* $O(|E| + k^{\omega} \log\log(n))$.

We note that the previous best known algorithm was obtained by Cheung et. al. [CKL13] and our corollary follows from using Theorem 33 in their reduction.

Next, we consider the Linear Matroid Union problem, where we are given as input a $n \times d$ matrix A such that $d \leqslant n$ and the goal is to find the largest number a disjoint basis. A basis is defined by a set of maximum number of linearly independent rows and two basis are disjoint if they do not share any rows. A special case of this problem includes computing the maximum number of edge-disjoint spanning trees.

Corollary 39 (Linear Matroid Union) *Given a* $n \times d$ *matrix* A *and* $k \in [n]$ *such that* $d \leqslant n$ *and* $\mathit{rank}(A) = r$, *there exists an algorithm that with probability* $99/100$ *outputs* $\min(k, \mathit{opt})$ *disjoint basis for* A. *Further, the algorithm runs in time* $O(k \operatorname{nnz}(A) + \min(r^{\omega} k^{\omega+1}, b^3 k^3) \log\log(n))$ *time.*

5 Matrix-vector-product sampling

We will use data structures $\textsc{Samp}(A, \Lambda, \mu_s)$ and $\textsc{DynSamp}(A, m_T, m_H)$, defined below, to sample from products of vectors (or matrices) with matrix A.

First, a simple folk-lore data structure.

Lemma 40 *Given* ℓ *real values* u_i, *there is a data structure using storage* $O(\ell)$ *such that a random* i *can be chosen with probability* $u_i^2 / \sum_{i'} u_{i'}^2$, *in time* $O(\log \ell)$. *Values can be inserted, deleted, or changed in the data structure in time* $O(\log \ell)$.

Proof: The data structure is simply a complete binary tree with each nonzero weight u_i^2 stored at a leaf, and with internal nodes recording the sum of the weights of their subtree. By choosing a path from the root at random, picking each left or right subtree with probability proportional to their weight, a leaf can be chosen with probability proportional to its weight. This data structure can be updated in $O(\log \ell)$ time, updating by walking from an updated leaf to the root, changing weights along the way. The storage is $O(\ell)$ since the complete binary tree has ℓ leaves. ■

Definition 41 *The members of* $\textsc{Samp}(A, \Lambda, \mu_s)$, *for* $A \in \mathbb{R}^{n \times d}$, Λ *a column selector matrix such that* $k = \mathit{rank}(A\Lambda) = \mathit{rank}(A)$, *and* $\mu_s \geqslant 1$, *are:*

- SA, *where* $S \in \mathbb{R}^{m_S \times n}$ *is a three-stage sketching matrix as in Corollary 17, with* $m_S = O(k/\varepsilon_0^2)$, *chosen to be an* ε_0*-embedding with failure probability* $1/k^{c_0}$, *for fixed* ε_0;
- C, *where* $Q, C \leftarrow \mathrm{QR}(SA\Lambda)$, *that is,* $SA\Lambda = QC$, Q *has orthonormal columns and* C *is triangular and invertible (since* $A\Lambda$ *has full column rank);*

- C_0, *where* $Q_0, C_0 \leftarrow \mathrm{QR}(SA)$*;*
- *The data structure of Lemma 40, built to enable sampling* $i \in [n]$ *with probability* $p_i \leftarrow \|Z_{i,*}\|^2/\|Z\|_F^2$ *in* $O(\log n)$ *time. Here* $Z \leftarrow A\Lambda(C^{-1}G)$*, with* $G \in \mathbb{R}^{k\times m_G}$ *having independent* $\mathcal{N}(0, 1/m_G)$ *entries, and* $m_G = \Theta(\mu_s)$*;*
- $\mu_s \geqslant 1$*, as input.*

Lemma 42 *The data structure* $\textsc{Samp}(A, \Lambda, \mu_s)$ *can be constructed in* $O(\mu_s(\mathtt{nnz}(A) + k^2) + d^\omega)$ *time.*

Proof: The time needed to compute SA is $O(\mu_s \,\mathtt{nnz}(A) + \varepsilon_0^{-2} dk^{1+1/\mu} \log^2(k/\varepsilon_0))$, for $\mu \geqslant 1$; we will take μ to be a constant large enough that the second term is dominated by other terms.

Computing the QR factorization of SA takes $O(d^\omega)$ time, by first computing $(SA)^\top(SA)$ for the $m_S \times d$ matrix SA, and then its Cholesky composition, using "fast matrix" methods for both, and using $m_S \leqslant d$. This dominates the time for the similar factorization of $SA\Lambda$.

The Z matrix can be computed in $O(\mu_s(\mathtt{nnz}(A) + k^2))$ time, by appropriate order of multiplication, and this dominates the time needed for building the data structure of Lemma 40.

Adding these terms, and using $m \leqslant \mathtt{nnz}(A)$, the result follows. ■

Algorithm MatVecSampler uses $\textsc{Samp}(A, \Lambda, \mu_s)$ to sample rows of AW with probability proportional to their squared Euclidean norm. The matrix L encodes both the sample indices and the probability with which the index is chosen. This style of output will be useful below; the matrix L will be a subspace embedding, when W is chosen appropriately. We don't construct L explicitly as an $v \times n$ matrix, and so do not include any vn terms in runtimes or space.

Algorithm 6 $\textsc{MatVecSampler}(A, \textsc{Sampler}, W, v, \nu)$

Input: $A \in \mathbb{R}^{n\times d}$, data structure Sampler (Def. 41), $W \in \mathbb{R}^{d\times m_W}$, desired number of samples v, normalizer ν, where $\nu = \frac{1}{6kn^{1/\mu_s}}$ by default if unspecified

Output: $L \in \mathbb{R}^{v\times d}$, encoding v draws from $i \in [n]$ chosen with approx. probability $q_i \overset{def}{=} \|A_{i,*}W\|^2/\|AW\|_F^2$

```
1: $N \leftarrow \|C_0 W\|_F^2$, where $C_0$ is from Sampler
2: if $N == 0$:
3:     return Uniform$(v, n)$      // Alternatively, raise an exception here
4: $L \leftarrow 0_{v\times n}$, $z \leftarrow 0$
5: while $z < v$:
6:     Choose $i \in [n]$ with probability $p_i$ using Sampler
7:     $\tilde{q}_i \leftarrow \|A_{i,*}W\|^2/N$
8:     With probability $\nu\frac{\tilde{q}_i}{p_i}$, accept $i$: set $L_{z,i} = 1/\sqrt{v\tilde{q}_i}$; $z \leftarrow z + 1$
9: return L
```

Lemma 43 *Given constant* $c_0 > 1$ *and small enough constant* $\varepsilon_0 > 0$*, and* Sampler *for* A*, there is an event* $\mathcal{E}$ *holding with failure probability at most* $1/k^{c_0}$*, so that if* $\mathcal{E}$ *holds, then when called with* $\nu \leftarrow \frac{1}{6kn^{1/\mu_s}}$*, the probability is* $(1 \pm \varepsilon_0)q_i$ *that the accepted index in Step 8 of* MatVecSampler *is* $i \in [n]$*, where* $q_i \overset{def}{=} \|A_{i,*}W\|^2/\|AW\|_F^2$*. The time taken is* $O(m_W d(d + vkn^{1/\mu_s}))$*, where* $k = \mathit{rank}(A)$*.*

Proof: We need to verify that the quantity in question is a probability, that is, that $\nu\frac{\tilde{q}_i}{p_i} \in (0, 1)$, when $\nu = \frac{1}{6kn^{1/\mu_s}}$.

From Corollary 17, if $m_S = O(\varepsilon_0^{-2}k)$ for $\varepsilon_0 > 0$, then with failure probability $1/k^{c_0+1}$, S will be a subspace ε_0-embedding for $\mathrm{im}(A)$, that is, for $A\Lambda$ and for A, using $\mathsf{rank}(A) = k$. The event $\mathcal{E}$ includes the condition that S is indeed an ε_0-embedding. If this holds for S, then from standard arguments, $A\Lambda C^{-1}$ has singular values all in $1 \pm \varepsilon_0$, and $\|A_{i,*}\Lambda C^{-1}\|^2 = (1 \pm O(\varepsilon_0))\tau_i$, where again τ_i is the i'th leverage score, as discussed in Section 1.2.3.

(We have $\|A\Lambda C^{-1}x\| = (1 \pm \varepsilon_0)\|SA\Lambda C^{-1}x\| = (1 \pm \varepsilon_0)\|x\|$, for all x, since $SA = QC$.) This implies that for Z, G in the construction of $\textsc{Samp}(A, \Lambda, \mu_s)$,

$$\|Z\|_F^2 = \|A\Lambda C^{-1}G\|_F^2 \leqslant (1 + \varepsilon_0)\|G\|_F^2.$$

Since $m_G\|G\|_F^2$ is χ^2 with km_G degrees of freedom, with failure probability at most $\exp(-\sqrt{k\mu_s}/2)$ (using $m_G = \Theta(\mu_s)$), it is at most $3km_G$ ([LM00], Lemma 1), so $\|G\|_F^2 \leqslant 3k$ with that probability. Our event $\mathcal{E}$ also includes the condition that this bound holds. Thus under this condition, $\|Z\|_F^2 \leqslant 3(1 + \varepsilon_0)k$.

From Lemma 14 and the above characterization of τ_i, for the Z of $\textsc{Samp}(A, \Lambda, \mu_s)$, $\|Z_{i,*}\|^2 = \|A_{i,*}\Lambda C^{-1}G\|^2 \geqslant (1 - O(\varepsilon_0))\tau_i/n^{1/\mu_s}$.

Putting these together, we have

$$p_i = \frac{\|Z_{i,*}\|_F^2}{\|Z\|^2} \geqslant (1 - O(\varepsilon_0))\frac{\tau_i/n^{1/\mu_s}}{3k}. \tag{6}$$

Using the ε_0-embedding property of S,

$$\|C_0W\|_F^2 = \|Q_0C_0W\|_F^2 = \|SAW\|_F^2 = (1 \pm 2\varepsilon_0)\|AW\|_F^2, \tag{7}$$

and so, letting $A = UC_1$ for U with orthonormal columns, we have, for small enough ε_0,

$$(1 - 2\varepsilon_0)\tilde{q}_i \leqslant \frac{\|A_{i,*}W\|^2}{\|AW\|_F^2} = \frac{\|U_{i,*}C_1W\|^2}{\|UC_1W\|_F^2} = \frac{\|U_{i,*}C_1W\|^2}{\|C_1W\|_F^2} \leqslant \frac{\|U_{i,*}\|^2\|C_1W\|^2}{\|C_1W\|_F^2} \leqslant \tau_i.$$

Putting this bound with (6) we have

$$\frac{\tilde{q}_i}{p_i} \leqslant \frac{\tau_i/(1 - 2\varepsilon_0)}{(1 - O(\varepsilon_0))\tau_i/n^{1/\mu_s}3(1 + \varepsilon_0)k} \leqslant 3(1 + O(\varepsilon_0))kn^{1/\mu_s}.$$

so that $\nu\frac{\tilde{q}_i}{p_i} = \frac{1}{6kn^{1/\mu_s}}\frac{\tilde{q}_i}{p_i} \leqslant (1 + O(\varepsilon_0))/2 \leqslant 1$ for small enough ε_0.

Using (7) we have $\tilde{q}_i = (1 \pm 2\varepsilon_0)q_i$.

Thus the correctness condition of the lemma follows, for small enough ε_0.

Turning to time: the time to compute C_0W is $O(d^2m_W)$. Each iteration takes $O(\log n + dm_W)$, for choosing i and computing $\tilde{q}_i$, and these steps dominate the time. As usual for rejection sampling, for the given acceptance probability $\nu\frac{\tilde{q}_i}{p_i}$, the expected number of iterations is $O(\nu/\nu) = O(\nu kn^{1/\mu_s})$. Adding these expressions yields the expected time bound, folding a factor of $\log n$ in by adjusting μ_s slightly. ■

6 Leverage-score sampling via rejection sampling

The algorithms in this section call FindBasis (Algorithm 3), and the matrix-product samplers of Section 5, where the input W of those samplers is $C^{-1}G'$, with G' a matrix of Gaussians, and C a change-of-basis matrix such that, for example $A\Lambda C^{-1}$ has singular values close to one. Here Λ selects $\operatorname{rank}(A)$ linearly independent columns. (Or, with a different matrix, $AR\Lambda C^{-1}$ has singular values close to one, where $R^\top$ is a subspace ε_0-embedding, for small constant ε_0.) As a result, via standard arguments $\|A_i\Lambda C^{-1}G'\|^2$ is a good estimate of τ_i, the i'th leverage score of $A\Lambda$ or $AR\Lambda$. We have $\operatorname{im}(A\Lambda) = \operatorname{im}(AR\Lambda) = \operatorname{im}(A)$, (where the claim about AR follows using the embedding property of R), which implies that the leverage scores of these three matrices are the same.

Algorithm 7 LevSample$(A, \mu_s, f())$

Input: $A \in \mathbb{R}^{n\times d}$, $\mu_s \geqslant 1$ specifying runtime tradeoff, function $f(\cdot) \to \mathbb{Z}_+$ returns a target sample size (may be just a constant)

Output: Leverage score sketching matrix L, column selector Λ

```
$k, \Lambda \leftarrow$ FindBasis$(A, \mathtt{col})$, get $k = \operatorname{rank}(A)$    // Algorithm 3; $\Lambda$ selects a column basis of $A$
Construct Sampler $\leftarrow$ Samp$(A\Lambda, I, \mu_s)$, use $C$ from it;    // Definition 41
$W \leftarrow C^{-1}G'$, where $G' \in \mathbb{R}^{k\times m_{G'}}$ with ind. $\mathcal{N}(0, 1/m_{G'})$ entries    // $m_{G'} = \Theta(\log n)$
$L \leftarrow$ MatVecSampler$(A\Lambda,$ Sampler$, W, f(k, C), \nu = 1/6n^{1/\mu_s})$
        // Algorithm 6, sample size $f(k, C)$, normalizer $\nu$
return $L, \Lambda$
```

Theorem 44 *Given constant $c_0 > 1$ and small enough constant $\varepsilon_0 > 0$, there is an event $\mathcal{E}$ holding with failure probability at most $1/k^{c_0}$, so that if $\mathcal{E}$ holds, then when called with $\nu \leftarrow \frac{1}{6n^{1/\mu_s}}$, for each sample, the probability is $(1 \pm \varepsilon_0)\tau_i$ that the sample is $i \in [n]$.*

Proof: The correctness analysis uses the condition of the proof of Lemma 43, up to using $k = n$, and a different normalizer ν. This includes the condition that S is an ε_0-embedding, which implies by standard analysis, as mentioned above, that the leverage score $\tau_i \geqslant (1 - O(\varepsilon_0))\|A_i\Lambda C^{-1}\|^2$.

Note that since $A\Lambda$ input to the Samp() data structure builder has full column rank, C and C_0 in Sampler are the same.

Correctness requires that $\nu\tilde{q}_i/p_i \leqslant 1$ in MatVecSampler when it is called here, with $W = C^{-1}G'$. Here by Lemma 14, with failure probability $1/n^{c_1}$ for fixed c_1, G' has $\|A_i\Lambda C^{-1}G'\| = (1 \pm \varepsilon_0)\|A_i\Lambda C^{-1}\|$ for all $i \in [n]$. Using $\|Qx\| = \|x\|$ for all x, when Q is an orthonormal matrix, and including also in $\mathcal{E}$ the condition that $\|Q_{i,*}G'\| \geqslant (1-\varepsilon_0)\|Q_{i,*}\|$, we have $\|G'\|_F^2 = \|QG'\|_F^2 \geqslant (1-\varepsilon_0)k$. Therefore $\|C_0W\|_F^2 = \|G'\|_F^2 \geqslant (1-\varepsilon_0)k$. It follows that

$$\tilde{q}_i = \frac{\|A_{i,*}\Lambda W\|^2}{\|C_0 W\|_F^2} = \frac{\|A_{i,*}\Lambda C^{-1}G'\|^2}{(1-\varepsilon_0)k} \leqslant (1+3\varepsilon_0)\frac{\|A_{i,*}\Lambda C^{-1}\|^2}{k},$$

a factor of $1/k$ smaller than the general bound for $\tilde{q}_i$ in the proof of Lemma 43. The correctness in the use of a smaller normalizer ν follows. ■

Theorem 45 *Let* $k = \mathit{rank}(A)$*, and choose* $\mu_s \geqslant 1$*. Algorithm* 7 *(*$\textsc{LevSample}(A, \nu, \mu_s, f(\cdot))$*) uses space* $O(n + k^\omega \log\log(nd))$*, not counting the space to store* A*, and runs in time*

$$O(\mu_s \,\mathtt{nnz}(A) + k^\omega \log\log(nd) + k^2 \log n + \nu k n^{1/\mu_s}),$$

where ν *is the sample size. For* $\nu = \varepsilon^{-2} k \log k$*, this bound can be expressed as* $O(\mu_e \,\mathtt{nnz}(A) + k^\omega \log\log(n) + \varepsilon^{-2-1/\mu_e} k^2)$ *time, for* $\mu_e \geqslant 1$.

Note: we can get a slightly smaller running time by including a more rounds of rejection sampling: the first round of sampling needs an estimate with failure probability totaled over for all n rows, while another round would only need such a bound for $\nu n^{1/\mu_s}$ rows; this would make the bound $\nu^{1+1/\mu_s} k n^{1/\mu_s^2}$, which would be smaller when $\nu \ll n$. However, in the latter case, the term $\nu k n^{1/\mu_s}$ is dominated by the other terms anyway, for relevant values of the parameters. For example if $\nu \leqslant n$ and $\nu k \leqslant \mathtt{nnz}(A)$ don't hold, then sampling is not likely to be helpful. Iterating $\log\log n$ times, a bound with leading term $O(\mathtt{nnz}(A)(\log\log n + \log \nu))$ is possible, but doesn't seem interesting.

Proof: Step 2, building $\textsc{Samp}(A\Lambda, \mu_s)$, take $O(\mu_s(\mathtt{nnz}(A) + k^2) + k^\omega)$ time, with d in Lemma 42 equal to k here.

From Lemma 43, the running time of $\textsc{MatVecSampler}$ is $O(k^2 \log n + \nu k^2 (\log n) n^{1/\mu_s})$, mapping d of the lemma to k, m_W to $m_{G'} = O(\log n)$. However, since the normalizer ν is a factor of k smaller than assumed in Lemma 43, the runtime in sampling is better by that factor. Also, we subsume the second $\log n$ factor by adjusting μ_s.

The cost of computing $C^{-1} G'$ is $O(k^2 \log n)$; we have a runtime of

$$\begin{aligned} O(\mathtt{nnz}(A) + k^\omega \log\log(n)) + O(\mu_s(\mathtt{nnz}(A) + k^2) + k^\omega) + O(k^2 \log n + \nu k n^{1/\mu_s}) \\ = O(\mu_s \,\mathtt{nnz}(A) + k^\omega \log\log(d) + k^2 \log n + \nu k n^{1/\mu_s}), \end{aligned}$$

as claimed.

Finally, suppose $\nu = \varepsilon^{-2} k \log k$, as suffices for an ε-embedding. If $\nu k n^{1/\mu_s} \leqslant \mathtt{nnz}(A) + k^\omega$, then the bound follows. Suppose not. If $n \geqslant k^\omega$, then

$$\varepsilon^{-2} \geqslant n^{1-1/\mu_s}/k^2 \log(k) \geqslant n^{1-1/\mu_s - 2/\omega}/\log(n)$$

and so $\varepsilon^{-2} \geqslant n^\gamma$, for constant $\gamma > 0$, implying $\varepsilon^{-2/\mu_s \gamma'} \geqslant n^{1/\mu_s} \log n$, for constant $\gamma' < \gamma$. When $k^\omega \geqslant n$,

$$\varepsilon^{-2} \geqslant k^{\omega - 2 - 1/\omega\mu_s}/\log(k) \geqslant k^\gamma,$$

for a constant $\gamma > 0$, so that $\varepsilon^{-\omega/\mu_s \gamma'} \geqslant n^{1/\mu_s} \log k$, for a constant $\gamma' < \gamma$. Using μ_e, a constant multiple of μ_s, to account for constants, the result follows. ■

7 Low-rank least-squares regression

We now consider low-rank least-squares regression in the static case, with the goal of sharpening prior work with respect to storage and dependence on $\mathtt{nnz}(A)$.

Algorithm 8 LSREG(A, b, μ)

Input: $A \in \mathbb{R}^{n\times d}$ of rank k, $b \in \mathbb{R}^d$, $\mu \geqslant 1$ specifying runtime tradeoffs
Output: $x' \in \mathbb{R}^n$ with at most k nonzero entries

```
$k, \Lambda \leftarrow$ FINDBASIS$(A^\top)$    // Algorithm 3; $k = \mathrm{rank}(A)$, $\Lambda$ selects $k$ lin. ind. columns
Construct SAMP$(A\Lambda, I, 1)$, use $C$ from it;    // Definition 41, sampler not used
$y_0 \leftarrow 0_k$, $z \leftarrow U^\top b$    // where $U \overset{def}{=} A\Lambda C^{-1}$, left in factored form
Choose $\tau$ large enough, in $O(\log(\kappa(A)/\varepsilon))$    // use $C$ to estimate $\kappa(A)$
for $h \leftarrow 0, 1, 2, \dots, \tau - 1$:
    $y_{h+1} \leftarrow z + y_h - U^\top U y_h = z + G_U y_h$    // where $G_U = I_k - U^\top U$
return $C^{-1} y_T$
```

Theorem 46 (Static, High-Precision Regression) *Given $A \in \mathbb{R}^{n\times d}$ with rank$(A) = k$, and $b \in \mathbb{R}^n$, in $O(n + k^\omega \log\log(n))$ space (not counting storage of A), with failure probability $1/k^{c_0}$, a vector $x' \in \mathbb{R}^n$ with at most k nonzero entries, such that $\|x' - x^*\|^2 \leqslant \varepsilon\|x^*\|^2$, can be found in*

$$O(\mathtt{nnz}(A)\log(\kappa(A)/\varepsilon) + k^2\log(\kappa(A)/\varepsilon) + k^\omega \log\log(n))$$

time, or $\hat{O}(\mathtt{nnz}(A) + k^\omega)$, where $\hat{O}$ ignores all log factors.

LSREG(A, b, μ) (Alg. 8) has the claimed properties. In the algorithm, the matrix U is computed explicitly, but left in its given factored form. When U or $U^\top$ are applied to a vector, the appropriate factors are applied in turn.

Proof: The first two steps of Algorithm 8 are the same as the first two of Algorithm 7, reducing the number of columns to k, and yielding matrix C so that $A\Lambda C^{-1}$ is well-conditioned. This takes the space claimed, via Theorem 45, and time $O(\mathtt{nnz}(A) + k^\omega \log\log(d))$, inspecting the running times of the appropriate steps, as in the proof.

Correctness is implied by the condition that AC^{-1} has singular values close to one; for this, it is enough that S of the algorithm is a subspace ε_d-embedding with small constant ε_d, say, $\varepsilon_d = 0.1$; this holds with failure probability at most $1/k^{c_0}$, as in Lemma 9.

The pre-conditioner C^{-1} is then used in an iterative method, just as in the proof of Theorem 43 of [CW13]. The iterative method needs $\log(\kappa(A)/\varepsilon)$ iterations, as discussed below, and each iteration involves a constant number of matrix-vector multiplications using C^{-1}, A, and $A^\top$; these take $O(\mathtt{nnz}(A) + k^2)$ time. Multiplying by the number of iterations, and adding the time to compute C, the overall work is as claimed.

The analysis of this simple iteration is well known, but for convenience, we give it here. For simplicity of notation, we will assume that A is full rank, so that Λ is the identity, and omitted.

To show that $O(\log(\kappa(A)/\varepsilon))$ iterations are sufficient, we describe the algorithm in more detail. Let $U \overset{def}{=} AC^{-1}$, so that the singular values of U are all $1 \pm \varepsilon_d$ for some small constant ε_d. We have $A^+ = C^{-1}U^+$; we find A^+b by finding U^+b and then multiplying by C^{-1}. (We do not compute U, but multiply by U and $U^\top$ using A and C^{-1}.)

Richardson iterations involve a square matrix G and vector z: we let $y_0 := 0$, and $y_{h+1} := Gy_h + z$. With this we have $y_h = (I - G^h)(I - G)^{-1}z$, assuming the inverse exists. That is, if $\|G\|$ is small, then we are estimating $(I - G)^{-1}z$ with error exponentially decreasing in the number of iterations.

We will apply this algorithm with $G_U := I_k - U^\top U$; since the singular values of U are in $1 \pm \varepsilon_d$, $\|G_U\|$ is at most $3\varepsilon_d$. We apply the algorithm with G_U and $z = U^\top b$, to obtain an estimate $z_1 = (U^\top U)^{-1} U^\top b + e_1 = U^+ b + e_1$. The error e_1 has $\|e_1\| \leqslant \varepsilon_1 \|U^+ b\|$, after $O(\log(1/\varepsilon_1))$ iterations. We then have the estimate $C^{-1} z_1 = C^{-1}(U^+ b + e_1) = A^+ b + C^{-1} e_1$. Since $A^+ = C^{-1} U^+$ and U has largest singular value at least $1 + \varepsilon_d$, we have $\|C^{-1}\| \leqslant \|A^+\|(1 + \varepsilon_d)$, and so $\|C^{-1} e_1\| \leqslant 2\varepsilon_1 \|A^+\| \|U^+ b\|$ Choosing $\varepsilon_1^2 = \frac{1}{2}\varepsilon \|A^+ b\| / \|A^+\| \|U^+ b\|$, we have $\|C^{-1} e_1\|^2 \leqslant \varepsilon \|A^+ b\|^2$, as desired for the error bound. Finally, since $\|U^+ b\| \leqslant \|U U^+ b\| / (1 - \varepsilon_d) = \|A A^+ b\| / (1 - \varepsilon_d)$, we have $1/\varepsilon_1$ within a constant factor of $\|A^+\| \|A A^+ b\| / \|A^+ b\| \leqslant \kappa(A)$, as claimed. ■

7.1 Low-rank regression: multiple response, via sampling

In this section, we give an algorithm for finding a submatrix of A, with $\mathrm{poly}(k/\varepsilon)$ rows and columns, such that the submatrix can be used to solve least-squares problems for which A is the design matrix.

We can use the following lemma, likely well-known, to translate from prediction error to solution error.

Lemma 47 *Let* $\gamma_{A,b} \overset{def}{=} \frac{\|b\|}{\|AA^+ b\|}$. *Recall that* $\kappa(A) \overset{def}{=} \|A\| \|A^+\|$. *Suppose* $\tilde{x} \in \mathrm{im}\, A^\top$, *and for some* $\varepsilon_p \in (0, 1)$,

$$\|A\tilde{x} - b\|^2 \leqslant (1 + \varepsilon_p) \|\xi^*\|^2 \tag{8}$$

holds, where $\xi^* \overset{def}{=} Ax^* - b$. *Then*

$$\|\tilde{x} - x^*\| \leqslant 2\sqrt{\varepsilon_p} \|A^+\| \|\xi^*\| \leqslant 2\sqrt{\varepsilon_p} \sqrt{\gamma_{A,b}^2 - 1}\, \kappa(A) \|x^*\|. \tag{9}$$

This extends to multiple response regression using $\gamma_{A,B}^2 \overset{def}{=} \frac{\|B\|_F^2}{\|AA^+ B\|_F^2}$, *by applying column by column to* B, *and extends to ridge regression, that is,* $A_{(\lambda)}$ *with* $\hat{B} = \begin{bmatrix} B \\ 0_{d \times d'} \end{bmatrix}$, *as well.*

Note that $x \in \mathrm{im}(A^\top) = \mathrm{im}(V)$ is no loss of generality, because the projection $VV^\top x$ of x onto $\mathrm{im}(A^\top)$ has $AVV^\top x = Ax$ and $\|VV^\top x\| \leqslant \|x\|$. So $\mathrm{argmin}_x \|Ax - b\|^2 + \lambda \|x\|$ must be in $\mathrm{im}(A^\top)$ for λ arbitrarily close to zero, and $A^+ b \in \mathrm{im}(A^\top)$.

For the ridge problem $\min_x \|A_{(\lambda)} x - \hat{b}\|$, we have $\|\xi^*\|^2 = \|A_{(\lambda)} x^* - \hat{b}\| = \|Ax^* - b\|^2 + \lambda \|x^*\|^2$, and recalling from Lemma 10 that, when A has SVD $A = U\Sigma V^\top$, $A_{(\lambda)}$ has singular value matrix D^{-1}, where $D \overset{def}{=} (\Sigma^2 + \lambda I_k)^{-1/2}$, so that $\kappa(A_{(\lambda)})^2 = (\lambda + \sigma_1^2)/(\lambda + \sigma_k^2)$, where A has singular values $\sigma_1, \dots, \sigma_k$, with $k = \mathrm{rank}(A)$.

Proof: Since $x^* = A^+ b = A^\top (AA^\top)^+ b \in \mathrm{im}\, A^\top$, we have $\tilde{x} - x^* = A^\top z \in \mathrm{im}\, A^\top$, for some z. Since $A^+ AA^\top = A^\top$, we have $\tilde{x} - x^* = A^\top z = A^+ AA^\top z = A^+ A(\tilde{x} - x^*)$. From the normal equations for regression and the Pythagorean theorem,

$$\|A(\tilde{x} - x^*)\|^2 = \|A\tilde{x} - b\|^2 - \|Ax^* - b\|^2 \leqslant 4\varepsilon_p \|\xi^*\|^2,$$

using $\|A\tilde{x} - b\| \leqslant (1+\varepsilon_p)\|\xi^*\|$ and $\varepsilon_p < 1$. Therefore, using also submultiplicativity of the spectral norm,

$$\begin{aligned}\|\tilde{x} - x^*\|^2 &= \|A^+A(\tilde{x} - x^*)\|^2 \\ &\leqslant \|A^+\|^2\|A(\tilde{x} - x^*)\|^2 \\ &\leqslant \|A^+\|^2 4\varepsilon_p\|\xi^*\|^2, \end{aligned} \tag{10}$$

and the first inequality of (9) follows. For the second, we bound

$$\frac{\|\xi*\|^2}{\|x^*\|^2} = \frac{\|b\|^2 - \|AA^+b\|^2}{\|A^+b\|^2} = \frac{(\gamma_{A,b}^2 - 1)\|AA^+b\|^2}{\|A^+b\|^2} \leqslant (\gamma_{A,b}^2 - 1)\|A\|^2$$

so from (10), we have

$$\|\tilde{x} - x^*\|^2 \leqslant \|A^+\|^2 4\varepsilon_p\|\xi^*\|^2 \leqslant \|A^+\|^2 4\varepsilon_p\|x^*\|^2(\gamma_{A,b}^2 - 1)\|A\|^2,$$

and the second inequality of (9) follows, using the definition of $\kappa(A)$. ■

Theorem 48 *Given $A \in \mathbb{R}^{n\times d}$ with rank$(A) = k$, and $B \in \mathbb{R}^{n\times d'}$. There is an algorithm that builds a matrix Λ selecting k columns of A, and a scaled sampling matrix $L \in \mathbb{R}^{m_L\times n}$, with $m_L = O(k\log k + \varepsilon^{-1}k)$, such that with constant failure probability,*

$$\tilde{X} \leftarrow \operatorname{argmin}_{X\in\mathbb{R}^{k\times d'}} \|L(A\Lambda X - B)\|$$

has $\|\tilde{X} - (A\Lambda)^+B\|_F^2 \leqslant \varepsilon\|A^+\|^2\|AA^+B - B\|_F^2$. The time taken to find Λ and L is

$$O(\mu_e \operatorname{nnz}(A) + k^\omega \log\log(n) + k^2(\log n + \varepsilon^{-1})^{1+1/\mu_e})$$

and the space $O(n + k^\omega \log\log(n))$, for given $\mu_e \geqslant 1$. The space needed for storing Λ and L is $O(m_L)$. The time to compute $\tilde{X}$ is $\tilde{O}(m_L k^{\omega-1} + d'(m_L k + k^\omega)$.

Note that $(A\Lambda)^+B$ is an optimal solution to $\min_{X\in\mathbb{R}^{n\times d'}} \|AX - B\|_F$, since it follows from normal equations that $A\Lambda(A\Lambda)^+B = AA^+B$.

A key point here is that Λ and L do not depend on B; the bound applies if the columns of B are presented sequentially, and the corresponding columns of $\tilde{X}$ computed sequentially, under the condition that the columns of B are not generated based on $\tilde{X}$.

The error bound obtained by the approximate minimizer $\tilde{X}$ here is not per-column; this makes it not entirely comparable with a given per-column bound. Still, this is close to the kind of guarantee provided by [GLT18].

Algorithm 9 LowRankReg(A, μ_e, B)

```
Input: $A \in \mathbb{R}^{n\times d}$, $\mu_r, \mu_e \geqslant 1$ determining performance tradeoffs,, $B \in \mathbb{R}^{n\times d'}$
Output: Approximate least-squares solution $\tilde{X} \in \mathbb{R}^{d\times d'}$
Let $f(k_x, C)$ return $k_x(\log k_x + \varepsilon^{-1}\|C^+\|^2)$ to estimate $m_L = O(k_x(\log k_x + \varepsilon^{-1}\|(A\Lambda)^+\|^2))$
$L, \Lambda \leftarrow$ LevSample$(A, \alpha\mu_e, f())$;    // Algorithm 7; $\alpha > 0$ a fixed constant
$\tilde{X} \leftarrow \operatorname{argmin}_{X\in\mathbb{R}^{k\times d'}} \|LA\Lambda X - LB\|_F$
return $\tilde{X}$
```

Proof: [of Theorem 48] The algorithm is simply to call LevSample (Alg. 7) for A with appropriate parameters, picking a sample of appropriate size. One condition on the sample is that its size be larger than that size ν such that L is an ε_0-embedding with sufficiently high probability, where $\varepsilon_0 > 0$ is a sufficiently small constant. Per Lemma 9, that is $\nu = O(\varepsilon_0^{-2} k \log k)$.

It is also necessary that L allow fast low-error matrix product estimation, so that Theorem 36 of [CW13] applies; per Lemma 32 of [CW13], there is $\nu' = O(\varepsilon_p^{-1} k)$ such that this holds. By Theorem 36 of [CW13], under these conditions, $\|A\Lambda\tilde{X} - B\|_F^2 \leqslant (1+\varepsilon_p)\|AA^+B - B\|_F^2$. Therefore, applying Lemma 47 just above for each column of $\tilde{X}$ and corresponding column of B, with $\varepsilon_p = \varepsilon$, $\|\tilde{X} - (A\Lambda)^+B\|_F^2 \leqslant \varepsilon\|A^+\|^2\|AA^+B - B\|_F^2$, as desired.

The needed $m_L = \nu + \nu' = O(k(\log k + \varepsilon^{-1}))$, giving an overall time bound for finding Λ and L of

$$O(\mu_s \operatorname{nnz}(A) + k^\omega \log\log(n) + k^2 \log n + m_L k n^{1/\mu_s}), \tag{11}$$

where the last term

$$m_L k n^{1/\mu_s} = k(\log k + \varepsilon^{-1}))k n^{1/\mu_s} = k^2 n^{1/\mu_s}(\log k + \varepsilon^{-1}\|A^+\|^2).$$

As in the proof of Theorem 45, if $k^2(\log k)n^{1/\mu_s} \geqslant \operatorname{nnz}(A)$, then $k \geqslant n^\gamma$ for fixed $\gamma > (1 - 1/\mu_s)/2$, and so $k^2(\log k)n^{1/\mu_s} \leqslant k^\omega \log n$ for large enough fixed μ_s. If $k^2\varepsilon^{-1}n^{1/\mu_s} \geqslant \operatorname{nnz}(A) + k^\omega$, then $\varepsilon^{-1} \geqslant n^\gamma$, for a fixed $\gamma \in (0,1)$, so $k^2(\varepsilon^{-1})^{1+1/\mu_e} \geqslant k^2\varepsilon^{-1}n^{1/\mu_s}$, for some μ_e a constant multiple of μ_s.

The time to compute $\tilde{X}$ from $LA\Lambda$ and LB is based on the time to set up and solve the normal equations, although other methods could of course be used. The result follows. ■

When the goal is small residual error $\|A\tilde{X} - B\|_F$, not estimation of $\tilde{X}$ itself, we obtain simpler expressions, that are optimal for $\operatorname{rank}(A) = d$.

Theorem 49 *Given $A \in \mathbb{R}^{n\times d}$ with $\operatorname{rank}(A) = k$, and $B \in \mathbb{R}^{n\times d'}$, there is an algorithm finds $\tilde{X}$ such that $\|A\tilde{X} - B\|_F \leqslant (1+\varepsilon)\min_X \|Ax - B\|_F$, that for constant ε takes time*

$$O(\operatorname{nnz}(A) + \min(d^\omega, k^\omega(\log\log(n))) + k^2(\log n + d') + k^\omega\varepsilon^{-1}) + \tilde{O}(\varepsilon^{-1}kd'),$$

or for $d' = 1$, $O(\operatorname{nnz}(A) + \min(d^\omega, k^\omega(\log\log(n))) + k^2 \log n + k^\omega\varepsilon^{-1})$.

Proof: The algorithm is the same as Algorithm 9, with leverage score sample size of $m_L = O(k(\log k + \varepsilon^{-1})$, up to point of computing $\tilde{X}$.

To compute $\tilde{X}$ from $LA\Lambda$ and LB, we apply a SparseGaussian $G \in \mathbb{R}^{m_G \times m_L}$, with $m_G = O(k\varepsilon^{-1})$, in time $\tilde{O}(m_L(k + d')) = \tilde{O}(\varepsilon^{-1}k(k+d'))$, and then set up and solve the normal equations, in time $O(k^\omega\varepsilon^{-1} + k^2d')$. We note that by Lemma 25, the fast low-error matrix product estimation condition continues to hold. Up to the change in sample size, the time to compute L and Λ is (11); the μ_s term is chosen so that the last term, with the n^{1/μ_s} multiplier, is dominated by others.

The basis-finding done in the leverage-score sampling can simply do nothing if $\operatorname{rank}(A)\log\log(n) > d$, leading to the minimum term. The result follows. ■

7.2 Dynamic Ridge Regression

Here we consider an algorithm using a data structure that represents input matrix $A \in \mathbb{R}^{n \times d}$ is maintained under insertions and deletions (and changes) in $O(\log(nd))$ time, such that a sampling a row/column proportional to leverage-scores can be generated in time independent of $\mathtt{nnz}(A)$ and whose dependence on n and d is $O(\log(nd))$.

Definition 50 $\textsc{DynSamp}(A)$ *is a data structure that, for $A \in \mathbb{R}^{n \times d}$, comprises:*

- *For each row and column of A, the data structure of Lemma 40 for the nonzero entries of the row or column.*
- *For the rows of A, the data structure of Lemma 40 for the squared lengths of the rows, and the corresponding data structure for the columns.*
- *A data structure supporting access to the value of entry a_{ij} of AT_R in $O(1)$ time.*

Lemma 51 $\textsc{DynSamp}(A^\top)$ *and* $\textsc{DynSamp}(A)$ *can be constructed from each other in constant time, and they can be maintained under turnstile updates of A in $O(\log(nd))$ time. Using* $\textsc{DynSamp}(A)$, *rows can be chosen at random with row $i \in [n]$ chosen with probability $\|A_{i,*}\|^2/\|A\|_F^2$ in $O(\log(nd))$ time, and similarly for columns.*

If $R \in \mathbb{R}^{d \times m}$ is a sampling matrix, so that AR has columns that are each a multiple of a column of A, then rows can be sampled from AR using that data structure and $\textsc{DynSamp}(A)$ *in $O(m \log(nd))$ time, with the row $i \in [n]$ chosen with probability $\|(AR)_{i,*}\|^2/\|AR\|_F^2$.*

An analogous statement is true for row sampling matrices.

Proof: Use Lemma 40 for the first part.

For the second, with a matrix R, construct the data structure of Lemma 40 for the columns of AR, in $O(m \log(nd))$ time. To sample, pick $j \in [m]$ with probability $\|AR_{*,j}\|^2/\|AR\|_F^2$, using the newly constructed data structure. Then pick $i \in [n]$ with probability $(AR)_{ij^*}^2/\|(AR)_{*,j^*}\|^2$. Adding the probabilities across the choices of j, the probability of choosing index i is $\|AR_{i,*}\|^2/\|AR\|_F^2$, as claimed.

Once a row is chosen, the time to determine the corresponding row length $\|(AR)_{i,*}\|$ is $O(m \log(nd))$, finding each $(AR)_{ij}$ for $j \in [m]$ in $O(\log(nd))$ time. ■

We will regard $\textsc{DynSamp}(A^\top)$ and $\textsc{DynSamp}(A)$ as interchangeable, and use these observations for sampling rows of sampled submatrices AR and columns of LA.

We designate the algorithm of Lemma 13 as $\textsc{LenSqEmbed}$, Algorithm 10. It will be used in two ways: when the rank is unknown, a sample large enough to be an ε-embedding is returned, otherwise, a sample of size so that a leverage score sample of size ν is expected.

Algorithm 10 $\textsc{LenSqEmbed}(A, \textsc{DynSampler}, Z, \texttt{dim} = \texttt{row}, \hat{k} = \texttt{unknown}, \varepsilon = \varepsilon_0, \nu = 0)$

Input: $A \in \mathbb{R}^{n \times d}$, $\textsc{DynSampler} = \textsc{DynSamp}(A)$, Z with $Z \geqslant \|A^+\|$, $\texttt{dim}$ flags sampling rows or columns (default is $\texttt{row}$), $\hat{k} = \mathrm{rank}(A)$ if not $\texttt{unknown}$, ε error parameter (defaults to ε_0, a small constant), ν leverage-score sample size (unused if $\hat{k} == \texttt{unknown}$)
Output: $\hat{L} \in \mathbb{R}^{m_{\hat{L}} \times n}$

```
1: if $\texttt{dim==col}$
       $A \leftarrow A^\top$
2: if $\hat{k} == \texttt{unknown}$
       $m_{\hat{L}} = O(\varepsilon^{-2} Z^2 \|A\|_F^2 \log d)$
   else
       $m_{\hat{L}} = O(\frac{\nu}{\hat{k}} Z^2 \|A\|_F^2)$
3: Construct $\hat{L}$ from $m_{\hat{L}}$ samples of the rows of $A$ as in Def. 12 and DynSampler, and using $w$,
   adding row $e_i^\top / \sqrt{m_{\hat{L}} p_i}$ to $\hat{L}$ when sampling picked row $i$.
4: return $\hat{L}$
```

This simple data structure and sampling scheme will be used to solve ridge regression problems, via the following algorithm.

Algorithm 11 $\textsc{RidgeRegDyn}(\textsc{DynSampler}, B, \hat{\sigma}_k, \hat{\sigma}_1, \varepsilon, \lambda)$

Input: $\textsc{DynSamp}(A)$ (Def. 50) for $A \in \mathbb{R}^{n \times d}$, $B \in \mathbb{R}^{n \times d'}$, $\hat{\sigma}_k \leqslant 1/\|A^+\|$, $\hat{\sigma}_1 \geqslant \|A\|$, ε an error parameter, λ a ridge weight
Output: Data for approximate ridge regression solution $A^\top S^\top \tilde{X}$ where S is a sampling matrix

```
1: $Z_\lambda \leftarrow 1/\sqrt{\lambda + \hat{\sigma}_k^2}$, $\hat{\kappa} \leftarrow Z_\lambda \sqrt{\lambda + \hat{\sigma}_1^2}$, $Z \leftarrow 1/\hat{\sigma}_k$
2: $S \leftarrow \textsc{LenSqEmbed}(A, \textsc{DynSampler}, Z_\lambda, \texttt{row}, \texttt{unknown}, \varepsilon/\hat{\kappa})$
           // Alg. 10; $m_S = O(\varepsilon^{-2} \hat{\kappa}^2 Z_\lambda^2 \|A\|_F^2 \log(d))$ rows
3: $R \leftarrow \textsc{LenSqEmbed}(SA, \textsc{DynSamp}(A), Z_\lambda, \texttt{col}, m_S, \varepsilon, m_S \hat{m}_R)$
           // cf. Lem. 51; $\hat{m}_R = O(\varepsilon^{-2} \log m_S)$, $m_R = O(\hat{m}_R Z_\lambda^2 \|A\|_F^2)$
4: $\tilde{X} \leftarrow (SARR^\top A^\top S^\top + \lambda I_{m_S})^{-1} SB$     // Solve using conjugate gradient
5: return $\tilde{X}$, $S$     // approximate ridge regression solution is $A^\top S^\top \tilde{X}$
```

Theorem 52 *Suppose* $\textsc{DynSamp}(A)$ *is maintained for* $A \in \mathbb{R}^{n \times d}$. *For a given invocation of Algorithm 11, with small constant failure probability,* $\|A^\top S^\top \tilde{X} - X^*\|_F \leqslant \varepsilon(\|X^*\|_F \gamma^2 + \frac{1}{\sqrt{\lambda}} \|U_{\lambda,\perp} B\|_F)$. *Here* $X^* \overset{def}{=} \operatorname{argmin}_X \|AX - B\|_F^2 + \lambda \|X\|_F^2$; $\gamma_{A_{(\lambda)}, \hat{B}}$ *given in Lemma 47 and after, with*

$$1 + \gamma_{\hat{A}_{(\lambda)}, \hat{B}}^2 = \frac{\|B\|_F^2}{\|AX^*\|_F^2 + \lambda \|X^*\|^2};$$

and $\|U_{\lambda,\perp} B\|$ *is a projection of* B *onto the bottom* $m_S - p$ *left singular matrix of* SA*, with* λ *between* $\sigma_{p+1}(SA)$ *and* $\sigma_p(SA)$. *An entry* $(A^\top S^\top \tilde{X})_{ij}$ *for given* i, j *can be computed in*

$$O(m_S \log(nd)) = O(\varepsilon^{-2} \hat{\kappa}^2 \psi_\lambda (\log(nd))^2)$$

time. The time taken to compute $\tilde{X}$ *is*

$$\tilde{O}(d' \varepsilon^{-4} \hat{\kappa}^2 \psi_\lambda^2 \kappa_\lambda \log(d)).$$

Here $\psi_\lambda \overset{def}{=} \|A\|_F^2/(\lambda+\hat{\sigma}_k^2)$, $\hat{\kappa} \overset{def}{=} \sqrt{(\lambda+\hat{\sigma}_1^2)/(\lambda+\hat{\sigma}_k^2)}$, *and* $\kappa_\lambda \overset{def}{=} (\lambda+\sigma_1(A)^2)/(\lambda+\sigma_k(A)^2)$.

Proof: Let

$$X_1 \overset{def}{=} \operatorname{argmin}_{X\in\mathbb{R}^{d\times d'}} \|SAX - SB\|_F^2 + \lambda\|X\|_F^2.$$

We first show that

$$\|X_1 - X^*\|_F \leqslant \varepsilon\gamma_{A_{(\lambda)},\hat{B}}\|X^*\|_F, \tag{12}$$

which follows from Lemma 47, applied to $A_{(\lambda)}$ and $\hat{B}$, after showing that, for $\varepsilon_p \overset{def}{=} \varepsilon/\hat{\kappa}^2$, X_1 satisfies

$$\|AX_1 - B\|_F^2 + \lambda\|X_1\|_F^2 \leqslant (1+\varepsilon_p/4)\Delta_*, \text{ where } \Delta_* \overset{def}{=} \|AX^* - B\|_F^2 + \lambda\|X^*\|_F^2, \tag{13}$$

which in turn follows from Lemma 17 of [ACW17]. That lemma considers a matrix U_1, comprising the first n rows of the left singular matrix of $\hat{A}_{(\lambda)} \overset{def}{=} \left[\begin{smallmatrix} A \\ \sqrt{\lambda}I_d \end{smallmatrix}\right]$, noting that the ridge objective can be expressed as $\min_X \|\hat{A}_{(\lambda)}X - \left[\begin{smallmatrix} B \\ \mathbf{0} \end{smallmatrix}\right]\|_F^2$. The matrix $U_1 = U\Sigma D$ in our terminology, as in Lemma 10, so the observations of that lemma apply.

Lemma 17 of [ACW17] requires that S satisfies

$$\|U_1^\top S^\top S U_1 - U_1^\top U_1\| \leqslant 1/4, \tag{14}$$

and

$$\|U_1^\top S^\top S(B - AX^*) - U_1^\top(B - AX^*)\|_F \leqslant \sqrt{\varepsilon_p\Delta_*}. \tag{15}$$

We have $\|A_{(\lambda)}^+\|^2 = 1/(\lambda + 1/\|A^+\|^2) \leqslant Z_\lambda$. With the given call to LenSqEmbed to construct S, the number of rows sampled is $m_S = O(\varepsilon_p^{-1}Z_\lambda^2\|A\|_F^2\log(d))$ (ignoring the Z/Z_λ term), so the expected number of times that row i of A is sampled is, up to a factor of $O(\log d)$,

$$\varepsilon_p^{-1}Z_\lambda^2\|A\|_F^2\frac{\|A_{i,*}\|^2}{\|A\|_F^2} = \varepsilon_p^{-1}Z_\lambda^2\|A_{i,*}\|^2 \geqslant \varepsilon_p^{-1}\|(U_1)_{i,*}\|^2 = \varepsilon_p^{-1}\|U_1\|_F^2\frac{\|(U_1)_{i,*}\|^2}{\|U_1\|_F^2},$$

and so row i is sampled at least the expected number of times it would be sampled under $\varepsilon_p^{-1}\|U_1\|_F^2\log d$ rounds of length-squared sampling of U_1. As shown by Rudelson and Vershynin ([RV07], see also [KV17], Theorem 4.4), this suffices to have, with high probability, a bound on the normed expression in (14) of

$$\frac{\|U_1\|\|U_1\|_F}{\sqrt{\varepsilon_p^{-1}\|U_1\|_F^2}} = \sqrt{\varepsilon_p}\|U_1\| \leqslant \sqrt{\varepsilon_p},$$

so by adjusting constant factors in sample size, (14) holds, for small enough ε_p.

To show that (15) holds, we use the discussion of the basic matrix multiplication algorithm discussed in [KV17], Section 2.1, which implies that

$$E[\|U_1^\top S^\top S(B - AX^*) - U_1^\top(B - AX^*)\|_F^2] \leqslant \frac{\|U_1\|_F^2\|B - AX^*\|_F^2}{s}$$

where s is number of length-squared samples of U_1. Here $s = \varepsilon_p^{-1}\|U_1\|_F^2\log d$, so (15) follows with constant probability by Chebyshev's inequality, noting that $\|B - AX^*\|_F \leqslant \sqrt{\Delta_*}$.

Thus (14) and (15) hold, so that by Lemma 17 of [ACW17], (13) holds. We now apply Lemma 47, which with (13) and $\varepsilon_p = \varepsilon^2/\hat{\kappa}^2$, implies (12).

Next we show that the (implicit) returned solution is close to the solution of (13), that is,

$$\|A^\top S^\top \tilde{X} - X_1\|_F^2 \leqslant \varepsilon \|X_1\|_F^2. \tag{16}$$

This is implied by Theorem 2 of [CYD18], since $A^\top S^\top \tilde{X}$ is the output for $t = 1$ of their Algorithm 1. (Or rather, it is their output for each column of $\tilde{X}$ and corresponding column of B.) To invoke their Theorem 2, we need to show that their equation (8) holds, which per their discussion following Theorem 3, holds for ridge leverage score sampling, with $O(\varepsilon^{-2} \mathtt{sd}_\lambda \log \mathtt{sd}_\lambda)$ samples, which our given m_R yields.

When we invoke their Theorem 2, we obtain

$$\|A^\top S^\top \tilde{X} - X_1\|_F \leqslant \varepsilon(\|X_1\|_F + \frac{1}{\sqrt{\lambda}}\|U_{k,\perp} B\|_F). \tag{17}$$

Combining with (12) and using the triangle inequality, we have that, abbreviating $\gamma_{\hat{A}_{(\lambda)},\hat{B}}$ as γ, up to the additive $U_{k,\perp}$ term in (17), we have

$$\begin{aligned}
\|A^\top S^\top \tilde{X} - X^*\|_F &\leqslant \|A^\top S^\top \tilde{X} - X_1\|_F + \|X_1 - X^*\|_F \\
&\leqslant \varepsilon\|X_1\|_F + \varepsilon(\gamma^2 - 1)\|X^*\|_F \\
&\leqslant \varepsilon\left[\|X^*\|_F + \varepsilon(\gamma^2-1)\|X^*\|_F + \gamma\|X^*\|_F\right] \\
&= \varepsilon\|X^*\|_F[1 + (\gamma^2 - 1)(1+\varepsilon)] \\
&\leqslant \varepsilon\|X^*\|_F 2\gamma^2,
\end{aligned}$$

for small enough ε, and then finally,

$$\|A^\top S^\top \tilde{X} - X^*\|_F \leqslant \varepsilon(\|X^*\|_F \gamma^2 + \frac{1}{\sqrt{\lambda}}\|U_{k,\perp} B\|_F),$$

as claimed.

The time is dominated by that for computing $\hat{A}^{-1}SB$, where $\hat{A} \overset{def}{=} SARR^\top A^\top S^\top$, which we do via the conjugate gradient method. Via standard results (see e.g. [Vis15], Thm 1.1), in $O((T + m_S)\sqrt{\kappa(\hat{A})}\log(1/\alpha))d'$ time, where T is the time to compute the product of $\hat{A}$ with a vector, we can obtain $\tilde{X}$ with $\|\tilde{X} - \hat{A}^{-1}SB\|_{\hat{A}} \leqslant \alpha\|\hat{A}^{-1}SB\|_{\hat{A}}$, where the $\hat{A}$-norm is $\|x\|_{\hat{A}} = x^\top \hat{A} x$. Since S and R are (at least) constant-factor subspace embeddings, the singular values of SAR are within a constant factor of those of A, and so $\kappa(\hat{A})$ is within a constant factor of

$$\kappa(AA^\top + \lambda I) = (\lambda + \sigma_1(A)^2)/(\lambda + \sigma_1(A)^2) = \kappa_\lambda^2.$$

We have

$$\begin{aligned}
T = O(m_R m_S) &= \tilde{O}(\varepsilon^{-2} \log m_S Z_\lambda^2 \|A\|_F^2 \varepsilon^{-2} \hat{\kappa}^2 Z_\lambda^2 \|A\|_F^2 \log(d)) \\
&= \tilde{O}(\varepsilon^{-4} \hat{\kappa}^2 Z_\lambda^4 \|A\|_F^4 \log(d))
\end{aligned}$$

Our running time is $\tilde{O}(T\kappa_\lambda \log(1/\varepsilon))d'$, with T as above. Translating to the notation using ψ terms, the result follows. ■

8 Sampling from a low-rank approximation

In this section, we give data structures for sampling from a low-rank approximation, first for the static case where matrix A is given as input, and then the dynamic case where single-entry updates to A are allowed.

The data structure for the static case is next, then the sampling procedure, followed by the analysis of the construction of the data structure.

Definition 53 EntrySampler *comprises, given* $\mu_s, \mu_r \geqslant 1$, *the sketches* AT_R, SA, S_2AR, SAR_2, *and* S_2AR_2, *all two-stage sketching matrices with* $S \in \mathbb{R}^{m_H \times n}$, $R^\top \in \mathbb{R}^{m_H \times d}$, $R = T_R H_R$ *where* $T_R \in \mathbb{R}^{d \times m_T}$, *and also:*

- $W_1, Z_1 \leftarrow \operatorname{argmin}_{W \in \mathbb{R}^{m_H \times k}, Z \in \mathbb{R}^{k \times m_H}} \|S_2ARWZSAR_2 - S_2AR_2\|_F^2$;
- $k, \Lambda \leftarrow \text{FindBasisL}(AT_R, \mu_r)$; *and*
- $\text{Sampler} \leftarrow \text{Samp}(AT_R, \Lambda, \mu_s)$.

The EntrySampler data structure can be used to generate entries $\hat{a}_{ij}$ of matrix $\hat{A} \overset{def}{=} ARW_1Z_1SA$, where $\hat{A}$ is a good rank-k approximation to A. Given $j \in [d]$, $\hat{a}_{ij}$ is generated with probability proportional to $\hat{a}_{ij}^2/\|\hat{A}_{*,j}\|^2$. This is done by computing the $m_T \times 1$ matrix $W \leftarrow H_RW_1Z_1(SA)_{*,j}$, with the multiplication associating from right to left, and then calling $\text{MatVecSampler}(AT_R, \text{Sampler}, W, \nu)$, (Alg. 6), where ν samples are desired.

The key fact here is that $\hat{A}$ is in fact a good approximation to A. This is a known result, but its formal statement, and a detailed discussion of the time needed to find it, including the computation of the sketches, is given next. Recall that $[A]_k$ denotes the best rank-k approximation to A.

Theorem 54 (Following [ACW17], Thm. 59) *Given* $A \in \mathbb{R}^{n \times d}$, $\varepsilon_d \in (0, 1/2)$, *and* $k \leqslant \mathit{rank}(A)$, *there is an algorithm that runs in* $O(\text{nnz}(A)) + \tilde{O}(\varepsilon_d^{-7}k^3)$ *time and with constant success probability, finds sketching matrices* $S \in \mathbb{R}^{m_S \times n}$ *and* $R \in \mathbb{R}^{d \times m_R}$, *and matrices* $W_1 \in \mathbb{R}^{m_H \times k}$ *and* $Z_1 \in \mathbb{R}^{k \times m_H}$, *such that* $\|ARW_1Z_1SA - A\|_F \leqslant (1 + \varepsilon_d)\|A - [A]_k\|_F$. *Here* $m_H = \tilde{O}(\varepsilon_d^{-1}k)$. *The matrices* T_S *and* $T_R^\top$ *have one nonzero entry per column. We will use two-stage* $S = H_ST_S$ *and* $R = T_RH_R$,*, where* H_S *and* $H_R^\top$ *are SRHTs and* T_S *and* $T_R^\top$ *are sparse embeddings (as described as* $\hat{T}$ *in Lemma 9). The matrices* AR *and* SA *can be computed in* $O(\text{nnz}(A)) + \tilde{O}(\varepsilon_d^{-2}k^3)$ *time, and* $\max\{\text{nnz}(AT_R), \text{nnz}(T_SA)\} \leqslant \text{nnz}(A)$.

Proof: The statement in [ACW17] states bounds only in terms of $\text{poly}(k/\varepsilon_d)$, without being more specific about the dependence on k and ε_d. Here we very briefly outline the parts of the analysis needed from [ACW17] to specify the dependence.

The theorem statement says that the matrices S and R are two-stage sketching matrices. For correctness, S and R must be ε_d-embeddings for small constant ε_d, and support approximation matrix multiplication, up to an error parameter $\sqrt{\varepsilon_d/k}$. For this, $m_T = O(\varepsilon_d^{-1}k + k^2)$ and $m_H = \tilde{O}(\varepsilon_d^{-1}k)$ suffice, where $T_S, T_R^\top$ have m_T rows, and $H_S, H_R^\top$ have m_H rows, leading to the bound given for computing SA and AR. (See Corollary 15 of [ACW17].)

To find W_1 and Z_1 of the theorem, a sketched version $\min_{W,Z} \|S_2ARWZSAR_2 - S_2AR_2\|$ is solved, where S_2 and $R_2^\top$ are *affine embeddings*, of the form $S_2 = H_{S_2}\hat{T}_{S_2}$ and similarly for R_2, but where S_2 and $R_2^\top$ must support approximate matrix multiplication with error parameter $\varepsilon_d/\sqrt{m_H} = \varepsilon_d/\sqrt{k/\varepsilon_d} = \varepsilon_d^{3/2}/\sqrt{k}$, and be an $O(\varepsilon_d)$-embedding. For this, it is enough that $\hat{T}_{S_2}$ sketch to $O(\varepsilon_d^{-3}k^2)$ dimensions, and H_{S_2} sketch to $m_2 \overset{def}{=} \varepsilon_d^{-3}k$ dimensions, and similarly for R_2. (See Theorem 26 of [ACW17].) Here $T_{S_2}, T_{R_2}^\top$ have one nonzero entry per column.

The two-sided sketches S_2AR, SAR_2, and S_2AR_2 can all be computed in $O(\mathtt{nnz}(A)) + \mathrm{poly}(k/\varepsilon_d)$ time, by first applying their constituent sparse embeddings on both sides first, and then the second stages.

Once S_2AR, SAR_2, and S_2AR_2 are computed, there remains the cost of computing Z_1 and W_1, for matrices all of which are $\mathrm{poly}(k/\varepsilon_d)$. Reviewing the proof of Lemma 27 of [ACW17], the dominant cost is, in the notation here, $\tilde{O}(\varepsilon_d^{-7}k^3)$ time, due to multiplying $S_2AR_2 \in \mathbb{R}^{m_2 \times m_2}$ on either side by matrices in $\mathbb{R}^{\tilde{O}(\varepsilon_d^{-1}k) \times m_2}$ (or the transpose of such a matrix). This takes $\tilde{O}(m_2^2\varepsilon_d^{-1}k)$ time, or $\tilde{O}(\varepsilon_d^{-7}k^3)$. ■

Adding also the analysis of MatVecSampler, we have

Theorem 55 *A data structure can be built in* $O(\mu_s\, \mathtt{nnz}(A)) + \tilde{O}(\mu_s k^3(k + \varepsilon_d^{-7}))$ *time, and using* $O(\mathtt{nnz}(A) + \tilde{O}(k^2(k + \varepsilon_d^{-1}))$ *space, such that given* $j \in [d]$, *a random* $i \in [n]$ *can be returned in expected* $O((\varepsilon_d^{-1}k + k^2)^2 n^{1/\mu_s})$ *time, where* i *is chosen with probability* $\hat{a}_{ij}^2/\|\hat{A}_{*,j}\|^2$, *and* $\hat{A}$ *has* $\|A - \hat{A}\|_F^2 \leqslant (1 + \varepsilon_d)\|A - [A]_k\|_F^2$.

Note that with $\mu = \log n$, the n^{1/μ_s} term in query time is constant, at the expense of $O(\mathtt{nnz}(A) \log n)$ preprocessing.

Proof: From Lemma 43, we have a bound on the generation time of $O(\nu m_T^2 n^{1/\mu_s}) = O(\nu(\varepsilon_d^{-1}k + k^2)^2 n^{1/\mu_s})$, mapping d, m_W, and k of the lemma to $m_T = O(\varepsilon_d^{-1}k + k^2)$, 1, and again m_T. ■

8.1 Dynamic Sampling from a Low-Rank Approximation

Our algorithm for the dynamic case starts with DynSamp(A) (Def. 50), and builds sampling matrices for A on its way to a low-rank approximation to A that can be sampled.

Algorithm 12 $\textsc{BuildLowRankFactors}(\textsc{DynSampler}, k, \hat{\sigma}_k, \hat{\sigma}_k, \varepsilon, \tau)$

Input: $\textsc{DynSamp}(A)$ (Def. 50) for $A \in \mathbb{R}^{n \times d}$, k target rank, $\hat{\sigma}_k \leqslant 1/\|A^+\|$, $\hat{\sigma}_k \leqslant \sigma_k(A)$, ε an error parameter, τ estimate of $\|A - A_k\|_F^2$, where A_k is the best rank-k approximation to A
Output: Small matrix W and sampling matrices S and R
so that $\text{rank}(ARWSA) = k$ and $\|ARWSA - A\| \leqslant (1+\varepsilon)\|A - A_k\|$

```
$\lambda \leftarrow \tau/k$, $Z_\lambda \leftarrow 1/\sqrt{\lambda + \hat{\sigma}_k^2}$, $Z_k \leftarrow 1/\hat{\sigma}_k$
$S \leftarrow \textsc{LenSqEmbed}(A, \textsc{DynSampler}, Z_\lambda, \texttt{row}, k, 0, k\hat{m}_S)$
            // Alg. 10; here $\hat{m}_S = O(\varepsilon^{-2} \log k)$, $S$ has $m_S = O(\hat{m}_S Z_\lambda^2 \|A\|_F^2)$ rows
$R_1 \leftarrow \textsc{LenSqEmbed}(SA, \textsc{DynSamp}(A), Z_\lambda, \texttt{col}, k, 0, k\hat{m}_S)$
Apply Alg. 1 and Thm. 1 of [CMM17] to $SAR_1$, get col. sampler $R_2$    // $m_{R_2} = O(\varepsilon^{-2} k \log k)$
Apply Alg. 1 and Thm. 1 of [CMM17] to $SAR_1R_2$, get row sampler $S_2$    // $m_{S_2} = O(\varepsilon^{-2} k \log k)$
$V \leftarrow$ top-$k$ right singular matrix of $S_2SAR_1R_2$
$U, \_ \leftarrow \text{QR}(SAR_1R_2V)$    // $U$ has orthonormal columns, $SAR_1R_2V = UC$ for a matrix $C$
$R_3 \leftarrow \textsc{LenSqEmbed}(SA, \textsc{DynSamp}(A), Z_k, \texttt{col}, k, 0, \varepsilon^{-1} k\hat{m}_{R_3})$
            // Here $\hat{m}_{R_3} = O(\varepsilon_0^{-2} \log k + \varepsilon^{-1})$, and $m_{R_3} = O(\hat{m}_{R_3} \varepsilon^{-1} Z_k^2 \|A\|_F^2$
Let $f(k, C)$ be the function returning the value $m_{R_4} = O(\varepsilon_0^{-2} k \log k + \varepsilon^{-1} k)$
$R_4^\top \leftarrow \textsc{LevSample}((U^\top SAR_3)^\top, \log(m_{R_3}), f())$    // Alg. 7
$R \leftarrow R_3R_4$
$W \leftarrow (U^\top SAR)^+ U^\top$
return $W$, $S$, $R$
```

Our algorithm uses Projection-Cost Preserving sketches, please see Definition 26 above.

We need the following lemma, implied by the algorithm and analysis in Section 5.2 of [BWZ16]; for completeness we include a proof.

Lemma 56 *If $S \in \mathbb{R}^{m_S \times n}$ and R are such that SA is a PCP of $A \in \mathbb{R}^{n \times d}$, and SAR is a PCP of SA, for error ε and rank t, and $U \in \mathbb{R}^{m_S \times k}$ has orthonormal columns such that $\|(I - UU^\top)SAR\|_F \leqslant (1+\varepsilon)\|SAR - [SAR]_t\|$, then $Y^* = \text{argmin}_Y \|YU^\top SA - A\|_F$ has*

$$\|Y^* U^\top SA - A\|_F \leqslant (1 + O(\varepsilon))\|A - A_t\|_F. \tag{18}$$

We also have

$$\|U^\top SA\|_F^2 \geqslant \|A_t\|_F^2 - O(\varepsilon)\|A\|_F^2.$$

Proof: Note that for matrix Y, $Y(I - Y_t^+ Y_t) = (I - Y_t Y_t^+)Y$, and that $UU^\top SA$ is no closer to SA than is the projection of SA to the rowspace of $U^\top SA$, and that $UU^\top = (SAR)_t(SAR)_t^+$ we have

$$\begin{aligned}\|A - Y^* U^\top SA\|_F = \|A(I - (U^\top SA)^+ U^\top SA)\|_F &\leqslant (1+\varepsilon)\|SA(I - (U^\top SA)^+ U^\top SA)\|_F \\ &\leqslant (1+\varepsilon)\|(I - UU^\top)SA\|_F \\ &\leqslant (1+\varepsilon)^2\|(I - UU^\top)SAR\|_F \\ &\leqslant (1+\varepsilon)^3\|(I - (SAR)_t(SAR)_t^+)SAR\|_F \\ &\leqslant (1+\varepsilon)^3\|(I - (SA)_t(SA)_t^+)SAR\|_F \\ &\leqslant (1+\varepsilon)^4\|(I - (SA)_t(SA)_t^+)SA\|_F \\ &= (1+\varepsilon)^4\|SA(I - (SA)_t^+(SA)_t)\|_F \\ &\leqslant (1+\varepsilon)^4\|SA(I - A_t^+ A_t)\|_F \\ &\leqslant (1+\varepsilon)^5\|A(I - A_t^+ A_t)\|_F \\ &= (1+\varepsilon)^5\|A - A_t\|_F = (1+O(\varepsilon))\|A - A_t\|_F,\end{aligned}$$

as claimed.

For the last statement: we have $\|SA\|_F^2 \geqslant (1-\varepsilon)\|A\|_F^2$, since SA is a PCP, and by considering the projection of A onto the rowspans of blocks of t of its rows. We have also $\|SA - [SA]_t\|_F^2 \leqslant (1+\varepsilon)\|A - [A]_t\|^2$, using that SA is a PCP. Using these observations, we have

$$\begin{aligned}\|[SA]_t\|_F^2 &= \|SA\|_F^2 - \|SA - [SA]_t\|_F^2 \\ &\geqslant (1-\varepsilon)\|A\|_F^2 - (1+\varepsilon)\|A - [A]_t\|_F^2 \\ &= \|[A]_t\|_F^2 - \varepsilon(\|A\|_F^2 + \|A - [A]_t\|_F^2) \\ &\geqslant \|[A]_t\|_F^2 - 3\varepsilon\|A\|_F^2.\end{aligned}$$

Similarly, $\|[SAR]_t\|_F^2 \geqslant \|[SA]_t\|_F^2 - 3\varepsilon\|SA\|_F^2$, using that SAR is a PCP of SA. We then have, using these inequalities, the PCP properties, and the hypothesis for U, that

$$\begin{aligned}\|U^\top SA\|_F^2 &= \|UU^\top SA\|_F^2 \\ &= \|SA\|_F^2 - \|(I - UU^\top)SA\|_F^2 \\ &\geqslant (1-\varepsilon)\|SAR\|_F^2 - (1+\varepsilon)^2\|SAR - [SAR]_t\|_F^2 \\ &\geqslant \|[SAR]_t\|_F^2 - 4\varepsilon\|SAR\|_F^2 \\ &\geqslant (\|[SA]_t\|_F^2 - 3\varepsilon\|SA\|_F^2) - 4(1+\varepsilon)\varepsilon\|SA\|_F^2 \\ &\geqslant (\|[A]_t\|_F^2 - 3\varepsilon\|A\|_F^2) - 3\varepsilon(1+\varepsilon)\|A\|_F^2 - 4(1+\varepsilon)^2\varepsilon\|A\|_F^2 \\ &\geqslant \|[A]_t\|_F^2 - 13\varepsilon\|A\|_F^2,\end{aligned}$$

for small enough ε, and the last statement of the lemma follows. ■

Theorem 57 *Given* DynSamp(A) *for* $A \in \mathbb{R}^{n\times d}$*, target rank* k*,* $\hat{\sigma}_k \leqslant 1/\|A^+\|$*,* $\hat{\sigma}_k \leqslant \sigma_k(A)$*, error parameter* ε*, and estimate* τ *of* $\|A - A_k\|_F^2$*. We assume* $\|A_k\|_F^2 \geqslant \varepsilon\|A\|_F^2$*. Then* BuildLowRankFactors*, Algorithm 12, returns* W *of rank* k *and sampling matrices* S *and* R *such that* $\|ARWSA - A\|_F \leqslant (1 + O(\varepsilon))\|A - A_k\|_F$*, where* A_k *is the best rank-*k *approximation to* A*. The time taken to find* W*,* S*, and* R *is*

$$\tilde{O}(\varepsilon^{-6}k^3 + \varepsilon^{-4}\psi_\lambda(\psi_\lambda + k^2 + k\psi_k)),$$

where $\psi_\lambda \overset{def}{=} \|A\|_F^2/(\tau/k + \hat{\sigma}_k^2)$ *and* $\psi_k \overset{def}{=} \|A\|_F^2/\sigma_k(A)$. *Given* $j \in [d]$, $i \in [n]$ *can be generated with probability* $(ARWSA)_{ij}^2/\|ARWSA)_{*,j}\|^2$ *in expected time* $O(\|A\|_F^2/\hat{\sigma}_k^2 + m_R^2\kappa(A)^2)$.

Here if the assumption $\|A_k\|_F^2 \geqslant \varepsilon\|A\|_F^2$ does not hold, the trivial solution 0 satisfies the relative error target:

$$\|A - 0\|_F^2 \leqslant \frac{1}{1-\varepsilon}(\|A\|_F^2 - \|A_k\|_F^2) = \frac{1}{1-\varepsilon}\|A - A_k\|_F^2 \leqslant (1+2\varepsilon)\|A - A_k\|^2,$$

and we assume the resulting approximation is not worth sampling.

Proof: With parameters as given to construct S, it will have at least $m_S = O(\hat{m}_S Z_\lambda^2 \|A\|_F^2)$ rows, and constitute an effective $k\hat{m}_S = O(\varepsilon^{-2}k\log k)$ ridge-leverage score samples of the rows of A. We assume that the input τ is within a constant factor of $\|A - A_k\|_F^2$, so that $\lambda = \tau/k$ is within a constant factor of $\|A - A_k\|_F^2/k$. Theorem 6 of [CMM17] implies that under these conditions, SA will be a rank-k Projection-Cost Preserving (PCP) sketch of A with error parameter ε, a (k, ε)-PCP.

Similarly to S, R_1 will be a (column) rank-k PCP of SA, here using that the PCP properties of SA imply that $\|(S(A - A_k)\|_F^2 = (1 \pm \varepsilon)\|A - A_k\|_F^2$, and so the appropriate λ, and Z_λ, for SA are within constant factors of those for A. Let $\hat{A} \overset{def}{=} SAR_1$. Lemma 16 and Theorem 1 of [CMM17] imply that applying their Algorithm 1 to $\hat{A}$ yields $S_2 \in \mathbb{R}^{m_{S_2} \times m_S}$ so that $S_2\hat{A}$ is a (k, ε)-PCP for $\hat{A}$, and similarly $S_2\hat{A}R_2$ is a (k, ε)-PCP for $S_2\hat{A}$.

We apply Lemma 56 with $\hat{A}^\top$, $R_2^\top$, $S_2^\top$, and $V^\top$ in the roles of A, S, R, and U in the lemma. We obtain that $\tilde{Y} \overset{def}{=} (\hat{A}R_2V)^+\hat{A} = \operatorname{argmin}_Y \|\hat{A}R_2VY - \hat{A}\|_F$ has $\|\hat{A}R_2V\tilde{Y} - \hat{A}\|_F \leqslant (1 + O(\varepsilon))\|\hat{A} - \hat{A}_k\|_F$, that is, U as constructed in Algorithm 12 has $UU^\top\hat{A} = \hat{A}R_2V\tilde{Y}$, and therefore satisfies the conditions of Lemma 56 for A, S, R_1. This implies that $Y^* \overset{def}{=} A(U^\top SA)^+ = \operatorname{argmin}_Y \|YU^\top SA - A\|_F$ has

$$\|Y^*U^\top SA - A\|_F \leqslant (1 + O(\varepsilon))\|A - A_k\|_F. \tag{19}$$

It remains to solve the multiple-response regression problem $\min_Y \|YU^\top SA - A\|_F$, which we do more quickly using the samplers R_3 and R_4.

We next show that $R_3^\top$ is a subspace ε_0-embedding of $(U^\top SA)^\top$, and supports low-error matrix product estimation, so that Thm. 36 of [CW13] can be applied. Per Lemma 9 and per Lemma 32 of [CW13], $k\hat{m}_{R_3} = O(\varepsilon_0^{-2}k\log k + \varepsilon^{-1}k)$ leverage-score samples of the columns of $U^\top SA$ suffice for these conditions to hold.

To obtain $k\hat{m}_{R_3}$ leverage score samples, we show that $1/\varepsilon$ length-squared samples of the columns of SA suffice to contain one length-squared sample of $U^\top SA$, and also that $\|(U^\top SA)^+\| \leqslant 1/\hat{\sigma}_k$, using the input condition on $\hat{\sigma}_k$ that $\sigma_k(A) \geqslant \hat{\sigma}_k$, so that the given value of $Z = Z_k$ in the call to LenSqEmbed for R_3 is valid.

For the first claim, by hypothesis $\|A_k\|_F^2 \geqslant \varepsilon\|A\|_F^2$, and by adjusting constants, U as computed satisfies the conditions of Lemma 56 for some $\varepsilon' = \alpha\varepsilon$ for constant $\alpha > 0$, so by that lemma and by hypothesis

$$\begin{aligned}\|U^\top SA\|_F^2 &\geqslant \|A_k\| - O(\alpha\varepsilon)\|A\|_F^2 \\ &\geqslant \varepsilon(1 - O(\alpha))\|A\|_F^2 \\ &\geqslant \varepsilon(1 - O(\alpha))(1 - \varepsilon)\|SA\|_F^2,\end{aligned}$$

so adjusting constants, we have $\|U^\top SA\|_F^2 \geqslant \varepsilon\|SA\|_F^2$. Using that U has orthonormal columns, we have for $j \in [d]$ that $\|U^\top SA_{*,j}\|/\|U^\top SA\|_F^2 \leqslant \|SA_{*,j}\|/\varepsilon\|SA\|_F^2$, so the probability of sampling j using length-squared probabilities for SA is least ε times that for $U^\top SA$.

For the claim for the value of Z used for R_3, using the PCP properties of SA and SAR_1, we have

$$\sigma_k(U^\top SA) = \sigma_k(UU^\top SAR_1) = \sigma_k(SAR_1) \geqslant (1-\varepsilon)\sigma_k(SA) \geqslant (1-O(\varepsilon))\sigma_k(A).$$

so the number of length-squared samples returned by LenSqEmbed suffice.

So using Thm. 36 of [CW13], $\tilde{Y}_3 = \operatorname{argmin}_Y \|(YU^\top SA - A)R_3\|_F$ satisfies

$$\|\tilde{Y}_3 U^\top SA - A\|_F \leqslant (1+\varepsilon)\min_Y \|YU^\top SA - A\|_F \leqslant (1+O(\varepsilon))\|A - A_k\|_F,$$

where the last inequality follows from (19).

Similar conditions and results can be applied to direct leverage-score sampling of the columns of $U^\top SAR_3$, resulting in $\tilde{Y}_4 = \min_Y \|(YU^\top SAR_3 - AR_3)R_4\|_F$, where there is $m_{R_4} = O(\varepsilon_0^{-2} k \log k + \varepsilon^{-1}k)$ such that these conditions hold for R_4. This implies $\tilde{Y}_4$ is an approximate solution to $\min_Y \|(YU^\top SA - A)R_3\|_F$, and therefore $AR\tilde{Y}_4 = AR(U^\top SAR)^+$ has $\|AR\tilde{Y}_4 U^\top SA - A\|_F \leqslant (1 + O(\varepsilon))\|A - A_k\|_F$, as claimed. We have $W \leftarrow (U^\top SAR)^+ U^\top = \tilde{Y}_4 U^\top$, so the claimed output condition on $ARWSA$ holds.

Turning to the time needed, Lemma 16 and Theorem 1 of [CMM17] imply that the time needed to construct S_2 and R_2 is

$$O(m_{R_1} m_S + k^2 m_S) = O(m_S(m_S + k^2)) \tag{20}$$

The time needed to construct V from $S_2 SAR_1 R_2 \in \mathbb{R}^{m_{S_2} \times m_{S_2}}$ is

$$O(m_{S_2}^3) = \tilde{O}(\varepsilon^{-6} k^3) \tag{21}$$

The time needed to construct U from $V \in \mathbb{R}^{m_{R_2} \times k}$ and $SAR_1 R_2 \in \mathbb{R}^{m_S \times m_{R_2}}$ by multiplication and QR factorization is

$$O(k m_{R_2} m_S + k^2 m_S) = \tilde{O}(m_S k^2 \varepsilon^{-2}) \tag{22}$$

Computation of $U^\top SAR_3$ requires $O(k m_S m_{R_3})$ time, where $m_{R_3} = O(\hat{m}_{R_3} \varepsilon^{-1} Z_k^2 \|A\|_F^2$, and $\hat{m}_{R_3} = O(\log k + \varepsilon^{-1})$, that is,

$$\tilde{O}(m_S k \varepsilon^{-2} Z_k^2 \|A\|_F^2) \tag{23}$$

time.

Leverage-score sampling of the rows of $(U^\top SAR_3)^\top \in \mathbb{R}^{m_{R_3} \times k}$ takes, applying Theorem 45 and using $m_{R_4} = O(k(\log k + \varepsilon^{-1}))$, time at most

$$\begin{align} O(\log(m_{R_3}) k m_{R_3} + k^\omega \log\log(k m_{R_3}) + k^2 \log m_{R_3} + m_{R_4} k m_{R_3}^{1/\log(m_{R_3})}) \tag{24} \\ = \tilde{O}(k\varepsilon^{-2} Z_k^2 \|A\|_F^2 + k^\omega + k^2 \varepsilon^{-1}) \tag{25} \end{align}$$

Computation of $(U^\top SAR)^+$ from $U^\top SAR$ requires $O(k^2 m_{R_4}) = \tilde{O}(\varepsilon^{-1} k^3)$ time. (With notation that R has $m_R = m_{R_4}$ columns.) Given $(U^\top SAR)^+$, computation of $W = (U^\top SAR)^+ U^\top$ requires

$$O(k m_R m_S) = \tilde{O}(m_S k^2 \varepsilon^{-1}) \tag{26}$$

time.

Putting together (20),(22), (23), (26), (21), (24), we have

$$\begin{aligned}
&O(m_S(m_S+k^2)) + \tilde{O}(m_S k^2 \varepsilon^{-2}) + \tilde{O}(m_S k \varepsilon^{-2} Z_k^2 \|A\|_F^2) + \tilde{O}(m_S k^2 \varepsilon^{-1}) \\
&+ \tilde{O}(\varepsilon^{-6} k^3) + tO(k\varepsilon^{-2} Z_k^2 \|A\|_F^2 + k^\omega + k^2 \varepsilon^{-1}) \\
&= \tilde{O}(m_S(m_S + k^2\varepsilon^{-2} + k\varepsilon^{-2} Z_k^2 \|A\|_F^2) + \varepsilon^{-6} k^3)
\end{aligned}$$

Here $m_S = O(\hat{m}_S Z_\lambda^2 \|A\|_F^2) = \tilde{O}(\varepsilon^{-2} Z_\lambda^2 \|A\|_F^2)$, so the time is

$$\tilde{O}(\varepsilon^{-6} k^3 + \varepsilon^{-4} Z_\lambda^2 \|A\|_F^2 (Z_\lambda^2 \|A\|_F^2 + k^2 + k Z_k^2 \|A\|_F^2))$$

Queries as in the theorem statement for given $j \in [d]$ can be answered as in [Tan19], in the time given. Briefly: given j, let $v \leftarrow (WSA)_{*,j} \in \mathbb{R}^{m_R}$. Let $\hat{A}$ denote AR. Using LenSqEmbed, generate a sampling matrix S_3 with $O(Z^2 \|\hat{A}\|^2)$ rows, and estimate $\|\hat{A}v\|^2 \approx \beta_v \leftarrow \|S\hat{A}v\|^2$. Generate $i \in [n]$ via rejection sampling as follows. In a given trial, pick $j^* \in [d]$ with probability proportional to $\|\hat{A}_{*,j}\|^2 v_j^2$ using DynSamp(A), then pick $i \in [m_R]$ with probability $\hat{A}_{i,j^*}^2 / \|\hat{A}_{*,j^*}\|^2$. This implies that $i \in [n]$ has been picked with probability $p_i \overset{def}{=} \sum_j \hat{A}_{ij}^2 v_j^2 / \sum_j \|\hat{A}_{*,j}\|^2 v_j^2$. Now for a value $\alpha > 0$, accept with probability $q_i/\alpha p_i$, where $q_i \overset{def}{=} (\hat{A}_{i,*}v)^2/\beta_v$, otherwise reject. This requires $\alpha \geqslant q_i/p_i$. and takes expected trials α. So an upper bound is needed for

$$\frac{q_i}{p_i} = \frac{(\hat{A}_{i,*}v)^2}{\beta_v} \frac{\sum_j \|\hat{A}_{*,j}\|^2 v_j^2}{\sum_j \hat{A}_{ij}^2 v_j^2}.$$

We have $(\hat{A}_{i,*}v)^2 \leqslant m_R \sum_j \hat{A}_{ij}^2 v_j^2$ using Cauchy-Schwarz, and $\beta_v \approx \|\hat{A}v\|^2 \geqslant \|v\|^2/\|\hat{A}^+\|^2$, and also $\sum_j \|\hat{A}_{*,j}\|^2 v_j^2 \leqslant \|v\|^2 \max_j \|\hat{A}_{*,j}\|^2 \leqslant \|v\|^2 \|\hat{A}\|^2$. Putting this together $\alpha \geqslant m_R \|\hat{A}\|^2 \|\hat{A}^+\|^2 = O(m_R \kappa(A))$ will do. The work per trial is $O(m_R \log(nd))$,and putting that together with the time to compute β_v, the theorem follows. ■

Acknowledgments: D. Woodruff would like to thank support from the National Institute of Health (NIH) grant 5R01 HG 10798-2, a Simons Investigator Award, and Office of Naval Research (ONR) grant N00014-18-1-256. The work of K. Clarkson and L. Horesh was supported in part by the Air Force Research Laboratory (AFRL) under contract FA8750-C-18-0098.

References

[ACW17] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. Sharper bounds for regularized data fitting. In *RANDOM '17: 21st International Workshop on Randomization and Computation*, 2017. Full version at `https://arxiv.org/abs/1611.03225`.

[BBB+19] Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P Woodruff. A PTAS for ℓ_p-low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 747–766. SIAM, 2019.

[BCJ20] Ainesh Bakshi, Nadiia Chepurko, and Rajesh Jayaram. Testing positive semidefiniteness via random submatrices. *arXiv preprint arXiv:2005.06441*, 2020.

[BCK15] D. W. Berry, A. M. Childs, and R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809, 2015.

[BCW19] Ainesh Bakshi, Nadiia Chepurko, and David P Woodruff. Robust and sample optimal algorithms for psd low-rank approximation. *arXiv preprint arXiv:1912.04177*, 2019.

[BG13] Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the subsampled randomized Hadamard transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1301–1340, 2013.

[BJW19] Ainesh Bakshi, Rajesh Jayaram, and David P Woodruff. Learning two layer rectified neural networks in polynomial time. In *Conference on Learning Theory*, pages 195–268. PMLR, 2019.

[BKL+19] Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. Quantum SDP Solvers: Large Speed-Ups, Optimality, and Applications to Quantum Learning. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BLWZ19] Maria-Florina Balcan, Yi Li, David P Woodruff, and Hongyang Zhang. Testing matrix rank, optimally. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 727–746. SIAM, 2019.

[BPSW20] Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. *arXiv preprint arXiv:2006.11648*, 2020.

[BW18] Ainesh Bakshi and David Woodruff. Sublinear time low-rank approximation of distance matrices. In *Advances in Neural Information Processing Systems*, pages 3782–3792, 2018.

[BWZ16] Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 236–249, 2016.

[CD16] Iris Cong and Luming Duan. Quantum discriminant analysis for dimensionality reduction and classification. *New Journal of Physics*, 18(7):073011, 2016.

[CEM+15] Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172, 2015.

[CGL+20] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 387–400, 2020.

[CKL13] Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *Journal of the ACM (JACM)*, 60(5):31, 2013.

[Cla05] Kenneth L Clarkson. Subgradient and sampling algorithms for ℓ_1 regression. In *Symposium on Discrete Algorithms: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, volume 23, pages 257–266, 2005.

[CLM+15] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pages 938–942, 2019.

[CLW18] Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. *CoRR*, abs/1811.04852, 2018.

[CMM17] Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

[CNW15] Michael B Cohen, Jelani Nelson, and David P Woodruff. Optimal approximate matrix product in terms of stable rank. *arXiv preprint arXiv:1507.02268*, 2015.

[Coh16] Michael B Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 278–287. SIAM, 2016.

[CP15] Michael B Cohen and Richard Peng. L_p row sampling by Lewis weights. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 183–192. ACM, 2015.

[CW13] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *STOC*, 2013. Full version at `http://arxiv.org/abs/1207.6365`. Final version J. ACM, Vol 63, 2017, `http://doi.acm.org/10.1145/3019134`.

[CW15] Kenneth L Clarkson and David P Woodruff. Input sparsity and hardness for robust subspace approximation. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 310–329. IEEE, 2015.

[CYD18] Agniva Chowdhury, Jiasen Yang, and Petros Drineas. An iterative, sketching-based framework for ridge regression. In *International Conference on Machine Learning*, pages 989–998, 2018.

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[DMIMW12] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *The Journal of Machine Learning Research*, 13(1):3475–3506, 2012.

[DMM06] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *APPROX-RANDOM*, pages 316–326, 2006.

[DW20] Vedran Dunjko and Peter Wittek. A non-review of quantum machine learning: trends and explorations. *Quantum Views*, 4:32, 2020.

[GLT18] András Gilyén, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*, 2018.

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204, 2019.

[GST20] András Gilyén, Zhao Song, and Ewin Tang. An improved quantum-inspired algorithm for linear regression. *arXiv preprint arXiv:2009.07268*, 2020.

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[IVWW19] Piotr Indyk, Ali Vakilian, Tal Wagner, and David Woodruff. Sample-optimal low-rank approximation of distance matrices. *arXiv preprint arXiv:1906.00339*, 2019.

[JKL$^+$20] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. *arXiv preprint arXiv:2009.10217*, 2020.

[JSWZ20] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster LPs. *arXiv preprint arXiv:2004.07470*, 2020.

[KN12] Daniel M. Kane and Jelani Nelson. Sparser Johnson-Lindenstrauss transforms. In *SODA*, pages 1195–1206, 2012.

[KP16] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*, 2016.

[KV09] Ravi Kannan and Santosh S. Vempala. Spectral algorithms. *Found. Trends Theor. Comput. Sci.*, 4(3-4):157–288, 2009.

[KV17] Ravindran Kannan and Santosh Vempala. Randomized algorithms in numerical linear algebra. *Acta Numerica*, 26:95, 2017.

[LGZ16] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of data. *Nature communications*, 7(1):1–7, 2016.

[LLMM20] Hannah Lawrence, Jerry Li, Cameron Musco, and Christopher Musco. Low-rank toeplitz matrix estimation via random ultra-sparse rulers. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4796–4800. IEEE, 2020.

[LM00] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Ann. Statist.*, 28(5):1302–1338, 10 2000.

[LMP13] Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 127–136. IEEE, 2013.

[LMR14] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

[LW20] Yi Li and David Woodruff. Input-sparsity low rank approximation in Schatten norm. *arXiv preprint arXiv:2004.12646*, 2020.

[Mah11] Michael W. Mahoney. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123–224, 2011.

[MP12] Gary L. Miller and Richard Peng. Iterative approaches to row sampling. *CoRR*, abs/1211.2713, 2012.

[MW17] Cameron Musco and David P Woodruff. Sublinear time low-rank approximation of positive semidefinite matrices. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 672–683. IEEE, 2017.

[NN13] Jelani Nelson and Huy L Nguyên. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 ieee 54th annual symposium on foundations of computer science*, pages 117–126. IEEE, 2013.

[Rec11] Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(Dec):3413–3430, 2011.

[RML14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, 113:130503, Sep 2014.

[RT08] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008.

[Rud99] Mark Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60–72, 1999.

[RV07] Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *J. ACM*, 54(4), 2007.

[Sar06] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152, 2006.

[SW19] Xiaofei Shi and David P Woodruff. Sublinear time numerical linear algebra for structured matrices. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4918–4925, 2019.

[SWZ17] Zhao Song, David P Woodruff, and Peilin Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 688–701, 2017.

[SWZ19] Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2772–2789. Society for Industrial and Applied Mathematics, 2019.

[Tan19] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 217–228. ACM, 2019.

[vAG19] Joran van Apeldoorn and András Gilyén. Improvements in Quantum SDP-Solving with Applications. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Vis15] Nisheeth Vishnoi. Cargese lecture notes. `http://www.cs.yale.edu/homes/vishnoi/CargeseLectures.pdf`, 2015.

[Woo14] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[WW19] Ruosong Wang and David P Woodruff. Tight bounds for ℓ_p oblivious subspace embeddings. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1825–1843. SIAM, 2019.

[ZFF19] Zhikuan Zhao, Jack K. Fitzsimons, and Joseph F. Fitzsimons. Quantum-assisted gaussian process regression. *Phys. Rev. A*, 99:052331, May 2019.

A Omitted Proofs for Subsection 2

Proof: [Proof of Lemma 9] The only novel claim here is for the space needed by the two-stage sketching scheme: the most direct implementation computes TA and then HTA, taking more than the claimed space. However, TA can be computed column by column: if A is stored column-wise, this is immediate, and if A is stored row-wise, $O(n)$ space suffices: insert the first entry of each row in a data structure with the entries stored in sets, one set per column. For each column, process the corresponding set, and for each entry of the set, insert the next entry of its row in the data structure.

Given that TA is computed column by column, H can be applied to each column in turn, and the resulting part of the sketch stored; this requires $O(m_{HT}n)$ space, as claimed. ■

Proof: [Proof of Lemma 14]

See e.g. [DG03] for the first claim. The second claim follows from e.g. Lemma 40 of [CW15], or via use of Lemma 2.2 of [DG03].

■

B Omitted Proofs for Section 3

Proof: [Proof of Lemma 19] Consider the first coordinate of the vector HDy,

$$(Hy)_1 = \sum_{j \in [n]} H_{1,j} y_j = \sum_{j \in [n]} X_j$$

Observe, the X_j's are mean zero random variables and that $|X_j| \leqslant H_{1,j}|y_j| \leqslant \frac{|y_j|}{\sqrt{n}}$, which follows from the definition of a Hadamard matrix. By a Chernoff-Hoeffding bound, for any $\eta > 0$,

$$\Pr\left[\left|\sum_{j \in [n]} X_j\right| > \eta\right] \leqslant 2\exp\left(-\frac{\eta^2 n}{2\sum_{j \in [n]} y_j^2}\right) \tag{27}$$

So, setting $\eta = \frac{c\|y\|_2 \log(n/\delta)}{\sqrt{n}}$, we can bound the above probability by $\frac{\delta}{cn}$. Union bounding over the error event for each of the n coordinates of the gives the claim. ■

Proof: [Proof of Lemma 20] We can use stochastic dominance the random variable $\sum_j X_j$ where $X_j = G_{1,j} y_j 1_{1,j}$ is dominated by the random variable $\|y\|_2 g_1$ where $g_1 \sim N(0,1)$. Then, $\|Gy\|_2^2 <=$

$\sum_{i\in[k]} \|y\|_2^2 g_i^2$, where the g_i's are standard Gaussian. Therefore, by Massart and Laurant [LM00] concentration for Chi-Squared random variables,

$$\Pr\left[\sum_{i\in[k]} \|y\|_2^2 g_i^2 > (k+2\sqrt{kt}+2t)\|y\|_2^2\right] \leqslant \exp(-t)$$

So setting $t = \varepsilon^2 k/c$ suffices to conclude with probability at least $1-\exp(-\varepsilon^2 k/c)$,

$$\|Gy\|_2^2 \leqslant (1+\varepsilon)k\|y\|_2^2$$

Setting $k = \log(1/\delta)/\varepsilon^2$ and C_{ij} to be $\mathcal{N}(0, 1/k)$, we can get k/ε^2 rows suffice to show with probability $1-2^{-k}$, to get $\|Gy\|_2 \leqslant (1+\varepsilon)\|y\|_2^2$ for a fixed y. Thus for a fixed vector y, with high probability, the norm does not dilate by more than a $1+\varepsilon$ factor.

■

Proof: [Proof of Lemma 21] We partition the vector y into levels based on the magnitude of each coordinate. We then prove that for any level set that contributes an $(\varepsilon/\log(n))$-fraction to the ℓ_2^2 norm of y is well approximated by a row of G. Formally, for all $\ell \in [\log^{c_1}(n/\varepsilon)]$, let

$$L_\ell = \left\{ i \mid 2^{-(\ell+1)}\|y\|_\infty \leqslant |(y)_i| \leqslant 2^{-\ell}\|y\|_\infty \right\}$$

denote the ℓ-th level set for the vector y. The ℓ-th level set contributes if $|L_\ell| \cdot 2^{-(2\ell+2)}\|y\|_\infty^2 \geqslant \frac{\varepsilon}{4\log^{c_1}(n/\varepsilon)}\|y\|_2^2$. We first observe that it suffices to only consider level sets that contribute. Note, for any coordinate of y smaller than $\|y\|_\infty/2^{\log^{c_1}(n/\varepsilon)}$, the contribution to the ℓ_2^2 norm of y is at most

$$\frac{1}{2^{\log^{2c_1}(n/\varepsilon)}}\|y\|_\infty^2 \leqslant \frac{1}{2^{\log^{2c_1}(n/\varepsilon)}} \cdot \frac{c\log(n)}{n}\|y\|_2^2 < \left(\frac{\varepsilon}{n}\right)^{c_1} \cdot \frac{1}{n}\|y\|_2^2$$

Since there are at most n such coordinates, the total contribution to the ℓ_2^2 norm is at most $\frac{\varepsilon}{n}\|y\|_2^2$. Similarly, if the ℓ-th level set does not "contribute", we can bound the ℓ_2^2 norm of the corresponding indices in L_ℓ as follows:

$$|L_\ell| \cdot \frac{1}{2^{2\ell}}\|y\|_\infty^2 \leqslant \frac{\varepsilon}{\log^{c_1}(n/\varepsilon)}\|y\|_2^2$$

Since there are at most $\log^{c_1}(n/\varepsilon)$ level sets, the total ℓ_2^2 norm restricted to all the level sets that do not contribute is at most $\varepsilon\|y\|_2^2$.

Now, we show that the norm of each contributing level set is well approximated by a row of G. Let $G_{1|L_\ell}$ be the first row of G restricted to the indices in a fixed level set L. For all $i \in L_\ell$, let X_i be the indicator for $G_{1,i}$ being non-zero. Observe, $\mathbb{E}\left[\sum_{i\in L_\ell} X_i\right] = \frac{\log^c(n)|L_\ell|}{n}$, for some fixed constant c. By Chernoff,

$$\Pr\left[\sum_{i\in L_\ell} X_i \leqslant (1-\eta)\frac{\log^c(k)|L_\ell|}{n}\right] \leqslant \exp\left(-\frac{|L_\ell|\eta^2}{2}\right)$$

Setting $\eta = 0.5$ and recalling $|L_\ell| \geqslant \frac{n2^\ell}{\log^{O(1)}(n/\varepsilon)} \geqslant \frac{n}{\log^{O(1)}(n/\varepsilon)}$, with probability at least $1 - \exp(-n^{0.6})$, $G_{1|L_\ell}$ has $\Omega(|L_\ell|/\log^{c+2}(n/\varepsilon))$ non-zero entries, each distributed as $\mathcal{N}(0, \varepsilon^2/k)$. Therefore,

$$\langle G_{1|L_\ell}, y\rangle = \langle G_1, y_{|L_\ell}\rangle = \|y_{|L_\ell}\|_2 g$$

where $g \sim \mathcal{N}(0, 1/k)$. Let ζ_i be the event that $G_{i|L_\ell}$ has $\Omega(|L_\ell|/\log^{c+2}(n/\varepsilon))$ non-zeros. Union bounding over all $i \in [k/\varepsilon^2]$ and all $\ell \in [\log^{c_1}(n/\varepsilon)]$, with probability at least $1 - \exp(-\sqrt{n})$, simultaneously for all ℓ, and for all $i \in [k/\varepsilon^2]$, $G_{i|L_\ell}$ has $\Omega(|L_\ell|/\log^{c+2}(n))$ non-zero entries.

Let $L = \cup_\ell L_\ell$ denote the union of all contributing level sets. Since $\|y_{|L}\|_2^2 \geqslant (1-\varepsilon)\|y\|_2^2$, it suffices for G to contract the norm of $y_{|L}$ by at most $(1-\varepsilon)$. Further, since L_ℓ form a partition of L and $G_{i|L_\ell}$ has at least $|L_\ell|/\Omega(\log^{c+2}(n))$, it follows that with probability at least $1 - \exp(-\sqrt{n})$, $G_{i|L_\ell}$ has at least $\frac{|L|}{\log^{c+2}(n)}$. Further,

$$\sum_{i \in [k]} \langle G_i, y_{|L}\rangle^2 = \sum_{i \in [k]} \|y_{|L}\|_2^2 g_i^2$$

which is a scaled chi-squared random variable with k degrees of freedom. Then, by using the lower tail bound from Massart and Laurant [LM00], we have

$$\Pr\left[\sum_{i \in [k]} \|y_{|L}\|_2^2 g_i^2 \leqslant (k - 2\sqrt{kt})\|y_{|L}\|_2^2\right] \leqslant \exp\left(-t\right)$$

Setting $t = k/\varepsilon^2$ and $g_i \sim \mathcal{N}(0, \varepsilon^2/k)$, with probability at least $1 - \exp(-k/10)$, $\|Gy\|_2^2 \geqslant \|Gy_{|L}\|_2^2 \geqslant (1-\varepsilon)\|y_{|L}\|_2^2 \geqslant (1-2\varepsilon)\|y\|_2^2$. ■

Proof: [Proof of Theorem 18] By assumption for all $x \in \mathbb{R}^d$, $\|Ax\|_\infty \leqslant \sqrt{\frac{\log(n)}{n}}\|Ax\|_2$. Then, from Lemmas 20 and 21 it follows that for a fixed vector $x \in \mathbb{R}^d$, with probability $1 - \exp(-k/\varepsilon^2)$, $\|GAx\|_2^2 = (1 \pm \varepsilon)\|Ax\|_2^2$. We show that constructing a fine enough net over $\mathbb{R}^d$ and union bounding over all vectors in the net suffices.

Using standard reductions for subspace embeddings, it suffices to consider A to have orthonormal columns and $x \in \mathcal{S}^{d-1}$. Further, we know that a greedy construction for a γ-net implies that there exists a subset of N points on $\mathcal{S}^{d-1}$, denoted by T, such that for all $x \in \mathbb{R}^d$, there exists $x' \in T$ such that $\|x - x'\|_2 \leqslant \gamma$. Further, $N = O((4/\gamma)^d)$. Now, let T' denote the set $\{Ax \mid x \in T\}$. It is easy to see that for all x, there exists a $y \in T'$ such that $\|Ax - y\|_2 \leqslant \gamma$. Let α be a constant such that $\|\alpha(y - y_1)\| = 1$ and so $\alpha \geqslant \frac{1}{\gamma}$. Note $\alpha(y - y_1)$ is still in the column space of A. Let $y_2' \in T'$ such that $\|\alpha(y - y_1) - y_2'\|_2 \leqslant \gamma$. Then $\|y - y_1 - \frac{y_2'}{\alpha}\|_2 \leqslant \frac{\gamma}{\alpha} \leqslant \gamma^2$. Set $y_2 = \frac{y_2'}{\alpha}$ and repeat, we obtain y_i that for all i, $\|y - y_1 - \ldots - y_i\|_2 \leqslant \gamma^i$. By triangle inequality,

$$\begin{aligned}
\|Sy\|_2^2 &= \|S\sum_i y_i\|_2^2 \\
&= \sum_i \|Sy_i\|_2^2 + 2\sum_{i,j,i\neq i}\langle Sy_i, Sy_j\rangle \\
&= \sum_i \|y_i\|_2^2 + 2\sum_{i,j,i\neq i}\langle y_i, y_j\rangle + O(\varepsilon)\sum_{i,j}\|y_i\|_2\|y_j\|_2 \\
&= \sum_i \|y_i\|_2^2 + 2\sum_{i,j,i\neq i}\langle y_i, y_j\rangle \pm O(\varepsilon) \\
&= \|\sum_i y_i\|_2^2 \pm O(\varepsilon) \\
&= 1 \pm O(\varepsilon)
\end{aligned} \tag{28}$$

Since this holds for an arbitrary $y = Ax$ for unit x, by linearity it follows that $\forall x, \|SAx\|_2 = (1\pm\varepsilon)\|Ax\|_2$. ■

Proof: [Proof of Lemma 23] Using triangle inequality of Frobenius norm, we have

$$\begin{aligned}
\|A^TS_1^TS_2^TS_2S_1B - A^TB\|_F &\leqslant \|A^TS_1^TS_2^TS_2S_1B - A^TS_1^TS_1B\|_F + \|A^TS_1^TS_1B - A^TS_1^TS_1B\|_F \\
&\leqslant \varepsilon\|S_1A\|_F\|S_1B\|_F + \varepsilon\|A\|_F\|B\|_F \\
&\leqslant O(\varepsilon)\|A\|_F\|B\|_F
\end{aligned} \tag{29}$$

■

Definition 58 ($(\varepsilon,\delta,\ell)$-JL Property [KN12]) *A distribution on matrices $S \in \mathbb{R}^{k\times n}$ has the $(\varepsilon,\delta,\ell)$-JL Property if for all $x \in \mathbb{R}^n$ with $\|x\|_2 = 1$,*

$$\mathbb{E}\left[\left|\|Sx\|_2^2 - 1\right|^\ell\right] \leqslant \varepsilon^\ell \cdot \delta$$

Proof: [Proof of Lemma 25] It follows from Kane and Nelson [KN12] that any sketch G that satisfies (ε,δ,l)-JL property also satisfies Approximate Matrix Product. We prove this for $l = 2$. For a unit vector x,

$$\mathbb{E}\left[\|Gx\|_2^2 - 1\right] = \mathbb{E}\left[\langle Gx, Gx\rangle - 1\right] = \langle \mathbb{E}\left[G^TG\right]x, x\rangle - 1 = \|x\|_2^2 - 1 = 0 \tag{30}$$

where we use that $\mathbb{E}\left[(G^TG)_{i,i}\right] = \mathbb{E}\left[\sum_l G_{i,l}^2\right] = 1$ and $\mathbb{E}\left[(G^TG)_{i,j}\right] = \mathbb{E}\left[\sum_l G_{i,l}G_{j,l}\right] = 0$. Next, we compute the variance. Observe,

$$\mathbb{E}\left[\|Gx\|_2^4\right] = \mathbb{E}\left[(\sum_{i,i_2}(G^TG)_{i,i_2}x_i, x_{i_2})^2\right] = \mathbb{E}\left[\sum_{i,i_2,i_3,i_4}(G^TG)_{i,i_2}(G^TG)_{i_3,i_4}\, x_i, x_{i_2}, x_{i_3}, x_{i_4}\right] \tag{31}$$

We break the above expectation into cases. When $i_1 = i_2 = i_3 = i_4$, we get

$$\mathbb{E}\left[\sum_{i_1}(G^TG)^2_{i_1,i_1}x^4_{i_1}\right] = \sum_{i_1}\mathbb{E}\left[(\sum_l G^2_{i,l})^2\right]x^4_{i_1} = \sum_{i_1}\sum_l \mathbb{E}\left[G^4_{i,l}\right]x^4_{i_1} \leqslant \frac{\log^c(n)}{k}\|x\|_4^4 \tag{32}$$

When $i_1 = i_2$ and $i_3 = i_4$, but $i_1 \neq i_3$ we get

$$\mathbb{E}\left[\sum_{i_1,i_3} (G^T G)_{i_1,i_1} (G^T G)_{i_3,i_3} x_{i_1}^2 x_{i_3}^2\right] = \sum_{i_1,i_3} \mathbb{E}\left[(G^T G)_{i_1,i_1}\right] \mathbb{E}\left[(G^T G)_{i_3,i_3}\right] x_{i_1}^2 x_{i_3}^2 = \|x\|_2^4 - \|x\|_4^4 \quad (33)$$

where we used that $\mathbb{E}\left[(G^T G)_{i_1,i_1}\right] = I$. When $i_1 = i_3$ and $i_2 = i_4$, but $i_1 \neq i_2$, we get

$$\begin{aligned}
\mathbb{E}\left[\sum_{i_1,i_2} (G^T G)_{i_1,i_2}^2 x_{i_1}^2 x_{i_2}^2\right] &= \sum_{i_1,i_2} \mathbb{E}\left[(\sum_{l} G_{i_1,l} G_{i_2,l})^2\right] x_{i_1}^2 x_{i_2}^2 \\
&= \sum_{i_2,i_2} \mathbb{E}\left[\sum_{l} G_{i_1,l}^2 G_{i_2,l}^2\right] x_{i_1}^2 x_{i_2}^2 \\
&= \sum_{i_1,i_2} \sum_{l} \mathbb{E}\left[G_{i_1,l}^2\right] \mathbb{E}\left[G_{i_2,l}^2\right] x_{i_1}^2 x_{i_2}^2 \\
&\leqslant \frac{\log^c(n)}{k} \|x\|_4^4
\end{aligned} \quad (34)$$

In the remaining cases we get 0. Then setting $k = \log^c(n)\log(1/\delta)/\varepsilon^2$ suffices and the claim follows. ■

Proof: [Proof of Lemma 27] We follow the proof structure for Theorem 12 in [CEM+15]. We begin by observing their proof requires $O(\varepsilon)$-subspace embedding and $O(\varepsilon/\sqrt{k})$-Approximate Matrix Product. We note that C has $O(k^{1.01}/\varepsilon^2)$ rows and thus is a subspace embedding, PHD has $O(k/\varepsilon)$ rows, and G has $O(k/\varepsilon)$ rows as well. By composition of subspace embeddings being a subspace embedding, condition 1 is satisfied. Further, by Lemma 24 we know that OSNAP and SRHT satisfy Approximate Matrix Product. Using Lemma 25 SparseGaussian also satisfies Approximate Matrix Product. Next, observe that each matrix is a subspace embedding and thus preserves Frobenius norm. Therefore, the sketches can be composed and the resulting matrix sketch satisfies Approximate Matrix Product as well. Therefore, we can apply Lemma 23 to conclude that $GPHDCA$ is a projection cost preserving sketch. ■