Fast Approximation of Matrix Coherence and Statistical Leverage

Petros Drineas   DRINEP@CS.RPI.EDU
Malik Magdon-Ismail   MAGDON@CS.RPI.EDU
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180

Michael W. Mahoney   MMAHONEY@CS.STANFORD.EDU
Department of Mathematics
Stanford University
Stanford, CA 94305

David P. Woodruff   DPWOODRU@US.IBM.COM
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120

Abstract

The *statistical leverage scores* of a matrix A are the squared row-norms of the matrix containing its (top) left singular vectors and the *coherence* is the largest leverage score. These quantities are of interest in recently-popular problems such as matrix completion and Nyström-based low-rank matrix approximation as well as in large-scale statistical data analysis applications more generally; moreover, they are of interest since they define the key structural nonuniformity that must be dealt with in developing fast randomized matrix algorithms. Our main result is a randomized algorithm that takes as input an arbitrary $n \times d$ matrix A, with $n \gg d$, and that returns as output relative-error approximations to *all n* of the statistical leverage scores. The proposed algorithm runs (under assumptions on the precise values of n and d) in $O(nd \log n)$ time, as opposed to the $O(nd^2)$ time required by the naïve algorithm that involves computing an orthogonal basis for the range of A. Our analysis may be viewed in terms of computing a relative-error approximation to an *under*constrained least-squares approximation problem, or, relatedly, it may be viewed as an application of Johnson-Lindenstrauss type ideas. Several practically-important extensions of our basic result are also described, including the approximation of so-called cross-leverage scores, the extension of these ideas to matrices with $n \approx d$, and the extension to streaming environments.

Keywords: matrix coherence, statistical leverage, randomized algorithm

1. Introduction

The concept of *statistical leverage* measures the extent to which the singular vectors of a matrix are correlated with the standard basis and as such it has found usefulness recently in large-scale data analysis and in the analysis of randomized matrix algorithms (Velleman and Welsch, 1981; Mahoney and Drineas, 2009; Drineas et al., 2008). A related notion is that of *matrix coherence*, which has been of interest in recently popular problems such as matrix completion and Nyström-based low-rank matrix approximation (Candes and Recht, 2008; Talwalkar and Rostamizadeh, 2010). Defined

more precisely below, the statistical leverage scores may be computed as the squared Euclidean norms of the rows of the matrix containing the top left singular vectors and the coherence of the matrix is the largest statistical leverage score. Statistical leverage scores have a long history in statistical data analysis, where they have been used for outlier detection in regression diagnostics (Hoaglin and Welsch, 1978; Chatterjee and Hadi, 1986). Statistical leverage scores have also proved crucial recently in the development of improved worst-case randomized matrix algorithms that are also amenable to high-quality numerical implementation and that are useful to domain scientists (Drineas et al., 2008; Mahoney and Drineas, 2009; Boutsidis et al., 2009; Drineas et al., 2006b; Sarlós, 2006; Drineas et al., 2010b); see Mahoney (2011) for a detailed discussion. The naïve and best previously existing algorithm to compute these scores would compute an orthogonal basis for the dominant part of the spectrum of A, for example, the basis provided by the Singular Value Decomposition (SVD) or a basis provided by a QR decomposition (Golub and Loan, 1996), and then use that basis to compute diagonal elements of the projection matrix onto the span of that basis.

We present a randomized algorithm to compute relative-error approximations to every statistical leverage score in time qualitatively faster than the time required to compute an orthogonal basis. For the case of an arbitrary $n \times d$ matrix A, with $n \gg d$, our main algorithm runs (under assumptions on the precise values of n and d, see Theorem 2 for an exact statement) in $O(nd \log n / \varepsilon^2)$ time, as opposed to the $\Theta(nd^2)$ time required by the naïve algorithm. As a corollary, our algorithm provides a relative-error approximation to the coherence of an arbitrary matrix in the same time. In addition, several practically-important extensions of the basic idea underlying our main algorithm are also described in this paper.

1.1 Overview and Definitions

We start with the following definition of the *statistical leverage scores* of a matrix.

Definition 1 *Given an arbitrary $n \times d$ matrix A, with $n > d$, let U denote the $n \times d$ matrix consisting of the d left singular vectors of A, and let $U_{(i)}$ denote the i-th row of the matrix U as a row vector. Then, the* statistical leverage scores *of the rows of A are given by*

$$\ell_i = \left\| U_{(i)} \right\|_2^2,$$

for $i \in \{1, \ldots, n\}$; the coherence γ *of the rows of A is*

$$\gamma = \max_{i \in \{1, \ldots, n\}} \ell_i,$$

that is, it is the largest statistical leverage score of A; and the (i, j)-cross-leverage scores c_{ij} are

$$c_{ij} = \left\langle U_{(i)}, U_{(j)} \right\rangle,$$

that is, they are the dot products between the ith row and the jth row of U.

Although we have defined these quantities in terms of a particular basis, they clearly do not depend on that particular basis, but only on the space spanned by that basis. To see this, let P_A denote the projection matrix onto the span of the columns of A. Then,

$$\ell_i = \left\| U_{(i)} \right\|_2^2 = \left(U U^T \right)_{ii} = (P_A)_{ii}. \tag{1}$$

That is, the statistical leverage scores of a matrix A are equal to the diagonal elements of the projection matrix onto the span of its columns.[1] Similarly, the (i, j)-cross-leverage scores are equal to the off-diagonal elements of this projection matrix, that is,

$$c_{ij} = (P_A)_{ij} = \langle U_{(i)}, U_{(j)} \rangle. \tag{2}$$

Clearly, $O(nd^2)$ time suffices to compute all the statistical leverage scores exactly: simply perform the SVD or compute a QR decomposition of A in order to obtain *any* orthogonal basis for the range of A and then compute the Euclidean norm of the rows of the resulting matrix. Thus, in this paper, we are interested in algorithms that run in $o(nd^2)$ time.

Several additional comments are worth making regarding this definition. First, since $\sum_{i=1}^n \ell_i = \|U\|_F^2 = d$, we can define a probability distribution over the rows of A as $p_i = \ell_i/d$. As discussed below, these probabilities have played an important role in recent work on randomized matrix algorithms and an important algorithmic question is the degree to which they are uniform or nonuniform.[2] Second, one could also define leverage scores for the columns of a "tall" matrix A, but clearly those are all equal to one unless $n < d$ or A is rank-deficient. Third, and more generally, given a rank parameter k, one can define the *statistical leverage scores relative to the best rank-k approximation to A* to be the n diagonal elements of the projection matrix onto the span of A_k, the best rank-k approximation to A.

1.2 Our Main Result

Our main result is a randomized algorithm for computing relative-error approximations to every statistical leverage score, as well as an additive-error approximation to all of the large cross-leverage scores, of an arbitrary $n \times d$ matrix, with $n \gg d$, in time qualitatively faster than the time required to compute an orthogonal basis for the range of that matrix. Our main algorithm for computing approximations to the statistical leverage scores (see Algorithm 1 in Section 3) will amount to constructing a "randomized sketch" of the input matrix and then computing the Euclidean norms of the rows of that sketch. This sketch can also be used to compute approximations to the large cross-leverage scores (see Algorithm 2 of Section 3).

The following theorem provides our main quality-of-approximation and running time result for Algorithm 1.

Theorem 2 *Let A be a full-rank $n \times d$ matrix, with $n \gg d$; let $\varepsilon \in (0, 1/2]$ be an error parameter; and recall the definition of the statistical leverage scores ℓ_i from Definition 1. Then, there exists a randomized algorithm (Algorithm 1 of Section 3 below) that returns values $\tilde{\ell}_i$, for all $i \in \{1, \ldots, n\}$, such that with probability at least* 0.8,

$$\left| \ell_i - \tilde{\ell}_i \right| \le \varepsilon \ell_i$$

---

1. In this paper, for simplicity of exposition, we consider the case that the matrix A has rank equal to d, that is, has full column rank. Theoretically, the extension to rank-deficient matrices A is straightforward—simply modify Definition 1 and thus Equations (1) and (2) to let U be any orthogonal matrix (clearly, with fewer than d columns) spanning the column space of A. From a numerical perspective, things are substantially more subtle, and we leave this for future work that considers numerical implementations of our algorithms.

2. Observe that if U consists of d columns from the identity, then the leverage scores are extremely nonuniform: d of them are equal to one and the remainder are equal to zero. On the other hand, if U consists of d columns from a normalized Hadamard transform (see Section 2.3 for a definition), then the leverage scores are very uniform: all n of them are equal to d/n.

holds for all $i \in \{1, \ldots, n\}$. Assuming $d \le n \le e^d$, the running time of the algorithm is

$$O\left(nd\ln\left(d\varepsilon^{-1}\right) + nd\varepsilon^{-2}\ln n + d^3\varepsilon^{-2}\left(\ln n\right)\left(\ln\left(d\varepsilon^{-1}\right)\right)\right).$$

Algorithm 1 provides a relative-error approximation to all of the statistical leverage scores ℓ_i of A and, assuming $d\ln d = o\left(\frac{n}{\ln n}\right)$, $\ln n = o\left(d\right)$, and treating ε as a constant, its running time is $o(nd^2)$, as desired. As a corollary, the largest leverage score (and thus the coherence) is approximated to relative-error in $o(nd^2)$ time.

The following theorem provides our main quality-of-approximation and running time result for Algorithm 2.

Theorem 3 *Let A be a full-rank $n \times d$ matrix, with $n \gg d$; let $\varepsilon \in (0, 1/2]$ be an error parameter; let κ be a parameter; and recall the definition of the cross-leverage scores c_{ij} from Definition 1. Then, there exists a randomized algorithm (Algorithm 2 of Section 3 below) that returns the pairs $\{(i, j)\}$ together with estimates $\{\tilde{c}_{ij}\}$ such that, with probability at least 0.8,*

i. If $c_{ij}^2 \ge \frac{d}{\kappa} + 12\varepsilon\ell_i\ell_j$, then (i, j) is returned; if (i, j) is returned, then $c_{ij}^2 \ge \frac{d}{\kappa} - 30\varepsilon\ell_i\ell_j$.

ii. For all pairs (i, j) that are returned, $\tilde{c}_{ij}^2 - 30\varepsilon\ell_i\ell_j \le c_{ij}^2 \le \tilde{c}_{ij}^2 + 12\varepsilon\ell_i\ell_j$.

This algorithm runs in $O(\varepsilon^{-2}n\ln n + \varepsilon^{-3}\kappa d\ln^2 n)$ time.

Note that by setting $\kappa = n\ln n$, we can compute all the "large" cross-leverage scores, that is, those satisfying $c_{ij}^2 \ge \frac{d}{n\ln n}$, to within additive-error in $O\left(nd\ln^3 n\right)$ time (treating ε as a constant). If $\ln^3 n = o\left(d\right)$ the overall running time is $o(nd^2)$, as desired.

1.3 Significance and Related Work

Our results are important for their applications to fast randomized matrix algorithms, as well as their applications in numerical linear algebra and large-scale data analysis more generally.

Significance in theoretical computer science. The statistical leverage scores define the key structural nonuniformity that must be dealt with (i.e., either rapidly approximated or rapidly uniformized at the preprocessing step) in developing fast randomized algorithms for matrix problems such as least-squares regression (Sarlós, 2006; Drineas et al., 2010b) and low-rank matrix approximation (Papadimitriou et al., 2000; Sarlós, 2006; Drineas et al., 2008; Mahoney and Drineas, 2009; Boutsidis et al., 2009). Roughly, the best random sampling algorithms use these scores (or the generalized leverage scores relative to the best rank-k approximation to A) as an importance sampling distribution to sample with respect to. On the other hand, the best random projection algorithms rotate to a basis where these scores are approximately uniform and thus in which uniform sampling is appropriate. See Mahoney (2011) for a detailed discussion.

As an example, the CUR decomposition of Drineas et al. (2008) and Mahoney and Drineas (2009) essentially computes $p_i = \ell_i/k$, for all $i \in \{1, \ldots, n\}$ and for a rank parameter k, and it uses these as an importance sampling distribution. The computational bottleneck for these and related random sampling algorithms is the computation of the importance sampling probabilities. On the other hand, the computational bottleneck for random projection algorithms is the application of the random projection, which is sped up by using variants of the Fast Johnson-Lindenstrauss Transform (Ailon and Chazelle, 2006, 2009). By our main result, the leverage scores (and thus

these probabilities) can be approximated in time that depends on an application of a Fast Johnson-Lindenstrauss Transform. In particular, the random sampling algorithms of Drineas et al. (2006b), Drineas et al. (2008) and Mahoney and Drineas (2009) for least-squares approximation and low-rank matrix approximation now run in time that is essentially the same as the best corresponding random projection algorithm for those problems (Sarlós, 2006).

Applications to numerical linear algebra. Recently, high-quality numerical implementations of variants of the basic randomized matrix algorithms have proven superior to traditional deterministic algorithms (Rokhlin and Tygert, 2008; Rokhlin et al., 2009; Avron et al., 2010). An important question raised by our main results is how these will compare with an implementation of our main algorithm. More generally, density functional theory (Bekas et al., 2007) and uncertainty quantification (Bekas et al., 2009) are two scientific computing areas where computing the diagonal elements of functions (such as a projection or inverse) of very large input matrices is common. For example, in the former case, "heuristic" methods based on using Chebychev polynomials have been used in numerical linear algebra to compute the diagonal elements of the projector (Bekas et al., 2007). Our main algorithm should have implications in both of these areas.

Applications in large-scale data analysis. The statistical leverage scores and the scores relative to the best rank-k approximation to A are equal to the diagonal elements of the so-called "hat matrix" (Hoaglin and Welsch, 1978; Chatterjee and Hadi, 1988). As such, they have a natural statistical interpretation in terms of the "leverage" or "influence" associated with each of the data points (Hoaglin and Welsch, 1978; Chatterjee and Hadi, 1986, 1988). In the context of regression problems, the ith leverage score quantifies the leverage or influence of the ith constraint/row of A on the solution of the overconstrained least squares optimization problem $\min_x \|Ax - b\|_2$ and the (i, j)-th cross leverage score quantifies how much influence or leverage the ith data point has on the jth least-squares fit (see Hoaglin and Welsch, 1978; Chatterjee and Hadi, 1986, 1988, for details). When applied to low-rank matrix approximation problems, the leverage score ℓ_j quantifies the amount of leverage or influence exerted by the jth column of A on its optimal low-rank approximation. Historically, these quantities have been widely-used for outlier identification in diagnostic regression analysis (Velleman and Welsch, 1981; Chatterjee et al., 2000).

More recently, these scores (usually the largest scores) often have an interpretation in terms of the data and processes generating the data that can be exploited. For example, depending on the setting, they can have an interpretation in terms of high-degree nodes in data graphs, very small clusters in noisy data, coherence of information, articulation points between clusters, the value of a customer in a network, space localization in sensor networks, etc. (Bonacich, 1987; Richardson and Domingos, 2002; Newman, 2005; Jonckheere et al., 2007; Mahoney, 2011). In genetics, dense matrices of size thousands by hundreds of thousands (a size scale at which even traditional deterministic QR algorithms fail to run) constructed from DNA Single Nucleotide Polymorphisms (SNP) data are increasingly common, and the statistical leverage scores can correlate strongly with other metrics of genetic interest (Paschou et al., 2007; Mahoney and Drineas, 2009; Drineas et al., 2010a; Paschou et al., 2010). Our main result will permit the computation of these scores and related quantities for significantly larger SNP data sets than has been possible previously (Paschou et al., 2007; Drineas et al., 2010a; Paschou et al., 2010; Georgiev and Mukherjee, 2011).

Remark. Lest there be any confusion, we should emphasize our main contributions. First, note that statistical leverage and matrix coherence are important concepts in statistics and machine learning. Second, recall that several random sampling algorithms for ubiquitous matrix problems such as least-squares approximation and low-rank matrix approximation use leverage scores in a

crucial manner; but until now these algorithms were $\Omega(T_{SVD})$, where T_{SVD} is the time required to compute a QR decomposition or a partial SVD of the input matrix. Third, note that, in some cases, $o(T_{SVD})$ algorithms exist for these problems based on fast random projections. But recall that the existence of those projection algorithms *in no way implies* that it is easy or obvious how to compute the statistical leverage scores efficiently. Fourth, one implication of our main result is that those random sampling algorithms can now be performed *just as efficiently* as those random projection algorithms; thus, the solution for those matrix problems can now be obtained while preserving the identity of the rows. That is, these problems can now be solved just as efficiently by using actual rows, rather than the arbitrary linear combinations of rows that are returned by random projections. Fifth, we provide a generalization to "fat" matrices and to obtaining the cross-leverage scores. Sixth, we develop algorithms that can compute leverage scores and related statistics even in streaming environments.

1.4 Empirical Discussion of Our Algorithms

Although the main contribution of our paper is to provide a rigorous theoretical understanding of fast leverage score approximation, our paper does analyze the theoretical performance of what is meant to be a practical algorithm. Thus, one might wonder about the empirical performance of our algorithms—for example, whether hidden constants render the algorithms useless for data of realistic size. Not surprisingly, this depends heavily on the quality of the numerical implementation, whether one is interested in "tall" or more general matrices, etc. We will consider empirical and numerical aspects of these algorithms in forthcoming papers, for example, Gittens and Mahoney (2012). We will, however, provide here a brief summary of several numerical issues for the reader interested in these issues.

Empirically, the running time bottleneck for our main algorithm (Algorithm 1 of Section 3) applied to "tall" matrices is the application of the random projection Π_1. Thus, empirically the running time is similar to the running time of random projection based methods for computing approximations to the least-squares problem, which is also dominated by the application of the random projection. The state of the art here is the Blendenpik algorithm of Avron et al. (2010) and the LSRN algorithm of Meng et al. (2011). In their Blendenpik paper, Avron, Maymounkov, and Toledo showed that their high-quality numerical implementation of a Hadamard-based random projection (and associated least-squares computation) "beats LAPACK's[3] direct dense least-squares solver by a large margin on essentially any dense tall matrix," and they concluded that their empirical results "suggest that random projection algorithms should be incorporated into future versions of LAPACK" (Avron et al., 2010). The LSRN algorithm of Meng, Saunders, and Mahoney improves Blendenpik in several respects, for example, providing better handling of sparsity and rank deficiency, but most notably the random projection underlying LSRN is particularly appropriate for solving large problems on clusters with high communication cost, for example, it has been shown to scale well on Amazon Elastic Cloud Compute clusters. Thus, our main algorithm should extend easily to these environments with the use of the random projection underlying LSRN. Moreover, for both Blendenpik and LSRN (when implemented with a Hadamard-based random projection), the hidden constants in the Hadamard-based random projection are so *small* that the random projection algorithm (and thus the empirical running time of our main algorithm for approximating leverage

---

3. LAPACK (short for Linear Algebra PACKage) is a high-quality and widely-used software library of numerical routines for solving a wide range of numerical linear algebra problems.

scores) beats the traditional $O(nd^2)$ time algorithm for dense matrices as small as thousands of rows by hundreds of columns.

1.5 Outline

In Section 2, we will provide a brief review of relevant notation and concepts from linear algebra. Then, in Sections 3 and 4, we will present our main results: Section 3 will contain our main algorithm and Section 4 will contain the proof of our main theorem. Section 5 will then describe extensions of our main result to general "fat" matrices, that is, those with $n \approx d$. Section 6 will conclude by describing the relationship of our main result with another related estimator for the statistical leverage scores, an application of our main algorithm to the under-constrained least-squares approximation problem, and extensions of our main algorithm to streaming environments.

2. Preliminaries on Linear Algebra and Fast Random Projections

We will start with a review of basic linear algebra and notation, and then we will describe the Fast Johnson-Lindenstrauss Transform and the related Subsampled Randomized Hadamard Transform.

2.1 Basic Linear Algebra and Notation

Let $[n]$ denote the set of integers $\{1, 2, \ldots, n\}$. For any matrix $A \in \mathbb{R}^{n \times d}$, let $A_{(i)}$, $i \in [n]$, denote the i-th row of A as a row vector, and let $A^{(j)}$, $j \in [d]$ denote the j-th column of A as a column vector. Let $\|A\|_F^2 = \sum_{i=1}^n \sum_{j=1}^d A_{ij}^2$ denote the square of the Frobenius norm of A, and let $\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2$ denote the spectral norm of A. Relatedly, for any vector $x \in \mathbb{R}^n$, its Euclidean norm (or ℓ_2-norm) is the square root of the sum of the squares of its elements. The dot product between two vectors $x, y \in \mathbb{R}^n$ will be denoted $\langle x, y \rangle$, or alternatively as $x^T y$. Finally, let $e_i \in \mathbb{R}^n$, for all $i \in [n]$, denote the standard basis vectors for $\mathbb{R}^n$ and let I_n denote the $n \times n$ identity matrix.

Let the rank of A be $\rho \leq \min\{n, d\}$, in which case the "compact" or "thin" SVD of A is denoted by $A = U\Sigma V^T$, where $U \in \mathbb{R}^{n \times \rho}$, $\Sigma \in \mathbb{R}^{\rho \times \rho}$, and $V \in \mathbb{R}^{d \times \rho}$. (For a general matrix X, we will write $X = U_X \Sigma_X V_X^T$.) Let $\sigma_i(A), i \in [\rho]$ denote the i-th singular value of A, and let $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ denote the maximum and minimum singular values of A, respectively. The Moore-Penrose pseudoinverse of A is the $d \times n$ matrix defined by $A^\dagger = V\Sigma^{-1}U^T$ (Nashed, 1976). Finally, for any orthogonal matrix $U \in \mathbb{R}^{n \times \ell}$, let $U^\perp \in \mathbb{R}^{n \times (n-\ell)}$ denote an orthogonal matrix whose columns are an orthonormal basis spanning the subspace of $\mathbb{R}^n$ that is orthogonal to the subspace spanned by the columns of U (i.e., the range of U). It is always possible to extend an orthogonal matrix U to a full orthonormal basis of $\mathbb{R}^n$ as $[U \quad U^\perp]$.

The SVD is important for a number of reasons (Golub and Loan, 1996). For example, the projection of the columns of A onto the k left singular vectors associated with the top k singular values gives the best rank-k approximation to A in the spectral and Frobenius norms. Relatedly, the solution to the least-squares (LS) approximation problem is provided by the SVD: given an $n \times d$ matrix A and an n-vector b, the LS problem is to compute the minimum ℓ_2-norm vector x such that $\|Ax - b\|_2$ is minimized over all vectors $x \in \mathbb{R}^d$. This optimal vector is given by $x_{opt} = A^\dagger b$. We call a LS problem *overconstrained* (or *overdetermined*) if $n > d$ and *underconstrained* (or *underdetermined*) if $n < d$.

2.2 The Fast Johnson-Lindenstrauss Transform (FJLT)

Given $\varepsilon > 0$ and a set of points $x_1, \ldots, x_n$ with $x_i \in \mathbb{R}^d$, a ε-Johnson-Lindenstrauss Transform (ε-JLT), denoted $\Pi \in \mathbb{R}^{r \times d}$, is a projection of the points into $\mathbb{R}^r$ such that

$$(1-\varepsilon)\|x_i\|_2^2 \leq \|\Pi x_i\|_2^2 \leq (1+\varepsilon)\|x_i\|_2^2.$$

To construct an ε-JLT with high probability, simply choose every entry of Π independently, equal to $\pm\sqrt{3/r}$ with probability $1/6$ each and zero otherwise (with probability $2/3$) (Achlioptas, 2003). Let Π_{JLT} be a matrix drawn from such a distribution over $r \times d$ matrices.[4] Then, the following lemma holds.

Lemma 4 (Theorem 1.1 of Achlioptas (2003)) *Let $x_1, \ldots, x_n$ be an arbitrary (but fixed) set of points, where $x_i \in \mathbb{R}^d$ and let $0 < \varepsilon \leq 1/2$ be an accuracy parameter. If*

$$r \geq \frac{1}{\varepsilon^2}\left(12\ln n + 6\ln\frac{1}{\delta}\right)$$

then, with probability at least $1 - \delta$, $\Pi_{JLT} \in \mathbb{R}^{r \times d}$ is an ε-JLT .

For our main results, we will also need a stronger requirement than the simple ε-JLT and so we will use a version of the Fast Johnson-Lindenstrauss Transform (FJLT), which was originally introduced in Ailon and Chazelle (2006) and Ailon and Chazelle (2009). Consider an orthogonal matrix $U \in \mathbb{R}^{n \times d}$, viewed as d vectors in $\mathbb{R}^n$. A FJLT projects the vectors from $\mathbb{R}^n$ to $\mathbb{R}^r$, while preserving the orthogonality of U; moreover, it does so very quickly. Specifically, given $\varepsilon > 0$, $\Pi \in \mathbb{R}^{r \times n}$ is an ε-FJLT for U if

- $\left\|I_d - U^T\Pi^T\Pi U\right\|_2 \leq \varepsilon$, and

- given any $X \in \mathbb{R}^{n \times d}$, the matrix product ΠX can be computed in $O(nd\ln r)$ time.

The next lemma follows from the definition of an ε-FJLT, and its proof can be found in Drineas et al. (2006b) and Drineas et al. (2010b).

Lemma 5 *Let A be any matrix in $\mathbb{R}^{n \times d}$ with $n \gg d$ and $\mathbf{rank}(A) = d$. Let the SVD of A be $A = U\Sigma V^T$, let Π be an ε-FJLT for U (with $0 < \varepsilon \leq 1/2$) and let $\Psi = \Pi U = U_\Psi \Sigma_\Psi V_\Psi^T$. Then, all the following hold:*

$$\mathbf{rank}(\Pi A) = \mathbf{rank}(\Pi U) = \mathbf{rank}(U) = \mathbf{rank}(A) = d,$$

$$\left\|I - \Sigma_\Psi^{-2}\right\|_2 \leq \varepsilon/(1-\varepsilon), \text{ and} \tag{3}$$

$$(\Pi A)^\dagger = V\Sigma^{-1}(\Pi U)^\dagger. \tag{4}$$

---

4. When no confusion can arise, we will use Π_{JLT} to refer to this distribution over matrices as well as to a specific matrix drawn from this distribution.

2.3 The Subsampled Randomized Hadamard Transform (SRHT)

One can use a Randomized Hadamard Transform (RHT) to construct, with high probability, an ε-FJLT. Our main algorithm will use this efficient construction in a crucial way.[5] Recall that the (unnormalized) $n \times n$ matrix of the Hadamard transform $\hat{H}_n$ is defined recursively by

$$\hat{H}_{2n} = \begin{bmatrix} \hat{H}_n & \hat{H}_n \\ \hat{H}_n & -\hat{H}_n \end{bmatrix},$$

with $\hat{H}_1 = 1$. The $n \times n$ normalized matrix of the Hadamard transform is equal to

$$H_n = \hat{H}_n / \sqrt{n}.$$

From now on, for simplicity and without loss of generality, we assume that n is a power of 2 and we will suppress n and just write H. (Variants of this basic construction that relax this assumption and that are more appropriate for numerical implementation have been described and evaluated in Avron et al. (2010).) Let $D \in \mathbb{R}^{n \times n}$ be a random diagonal matrix with independent diagonal entries $D_{ii} = +1$ with probability $1/2$ and $D_{ii} = -1$ with probability $1/2$. The product HD is a RHT and it has three useful properties. First, when applied to a vector, it "spreads out" its energy. Second, computing the product HDx for any vector $x \in \mathbb{R}^n$ takes $O(n \log_2 n)$ time. Third, if we only need to access r elements in the transformed vector, then those r elements can be computed in $O(n \log_2 r)$ time (Ailon and Liberty, 2008). The Subsampled Randomized Hadamard Transform (SRHT) randomly samples (according to the uniform distribution) a set of r rows of a RHT.

Using the sampling matrix formalism described previously (Drineas et al., 2006a,b, 2008, 2010b), we will represent the operation of randomly sampling r rows of an $n \times d$ matrix A using an $r \times n$ linear sampling operator S^T. Let the matrix $\Pi_{FJLT} = S^T HD$ be generated using the SRHT.[6] The most important property about the distribution Π_{FJLT} is that if r is large enough, then, with high probability, Π_{FJLT} generates an ε-FJLT. We summarize this discussion in the following lemma (which is essentially a combination of Lemmas 3 and 4 from Drineas et al. (2010b), restated to fit our notation).

Lemma 6 *Let $\Pi_{FJLT} \in \mathbb{R}^{r \times n}$ be generated using the SRHT as described above and let $U \in \mathbb{R}^{n \times d}$ ($n \gg d$) be an (arbitrary but fixed) orthogonal matrix. If*

$$r \geq \frac{14^2 d \ln(40nd)}{\varepsilon^2} \ln\left(\frac{30^2 d \ln(40nd)}{\varepsilon^2}\right),$$

then, with probability at least 0.9, Π_{FJLT} is an ε-FJLT for U.

3. Our Main Algorithmic Results

In this section, we will describe our main results for computing relative-error approximations to every statistical leverage score (see Algorithm 1) as well as additive-error approximations to all of the large cross-leverage scores (see Algorithm 2) of an arbitrary matrix $A \in \mathbb{R}^{n \times d}$, with $n \gg d$. Both algorithms make use of a "randomized sketch" of A of the form $A(\Pi_1 A)^\dagger \Pi_2$, where Π_1 is an ε-FJLT and Π_2 is an ε-JLT. We start with a high-level description of the basic ideas.

---

5. Note that the RHT has also been crucial in the development of $o(nd^2)$ randomized algorithms for the general over-constrained LS problem (Drineas et al., 2010b) and its variants have been used to provide high-quality numerical implementations of such randomized algorithms (Rokhlin and Tygert, 2008; Avron et al., 2010).

6. Again, when no confusion can arise, we will use Π_{FJLT} to denote a specific SRHT or the distribution on matrices implied by the randomized process for constructing an SRHT.

3.1 Outline of Our Basic Approach

Recall that our first goal is to approximate, for all $i \in [n]$, the quantities

$$\ell_i = \left\| U_{(i)} \right\|_2^2 = \left\| e_i^T U \right\|_2^2, \tag{5}$$

where e_i is a standard basis vector. The hard part of computing the scores ℓ_i according to Equation (5) is computing an orthogonal matrix U spanning the range of A, which takes $O(nd^2)$ time. Since $UU^T = AA^\dagger$, it follows that

$$\ell_i = \left\| e_i^T UU^T \right\|_2^2 = \left\| e_i^T AA^\dagger \right\|_2^2 = \left\| (AA^\dagger)_{(i)} \right\|_2^2, \tag{6}$$

where the first equality follows from the orthogonality of (the columns of) U. The hard part of computing the scores ℓ_i according to Equation (6) is two-fold: first, computing the pseudoinverse; and second, performing the matrix-matrix multiplication of A and $A^\dagger$. Both of these procedures take $O(nd^2)$ time. As we will see, we can get around both of these bottlenecks by the judicious application of random projections to Equation (6).

To get around the bottleneck of $O(nd^2)$ time due to computing $A^\dagger$ in Equation (6), we will compute the pseudoinverse of a "smaller" matrix that approximates A. A necessary condition for such a smaller matrix is that it preserves rank. So, naïve ideas such as uniformly sampling $r_1 \ll n$ rows from A and computing the pseudoinverse of this sampled matrix will not work well for an arbitrary A. For example, this idea will fail (with high probability) to return a meaningful approximation for matrices consisting of $n-1$ identical rows and a single row with a nonzero component in the direction perpendicular to that the identical rows; finding that "outlying" row is crucial to obtaining a relative-error approximation. This is where the SRHT enters, since it preserves important structures of A, in particular its rank, by first rotating A to a random basis and then uniformly sampling rows from the rotated matrix (see Drineas et al., 2010b for more details). More formally, recall that the SVD of A is $U\Sigma V^T$ and let $\Pi_1 \in \mathbb{R}^{r_1 \times n}$ be an ε-FJLT for U (using, for example, the SRHT of Lemma 6 with the appropriate choice for r_1). Then, one could approximate the ℓ_i's of Equation (6) by

$$\hat{\ell}_i = \left\| e_i^T A (\Pi_1 A)^\dagger \right\|_2^2, \tag{7}$$

where we approximated the $n \times d$ matrix A by the $r_1 \times d$ matrix $\Pi_1 A$. Computing $A(\Pi_1 A)^\dagger$ in this way takes $O(ndr_1)$ time, which is not efficient because $r_1 > d$ (from Lemma 6).

To get around this bottleneck, recall that we only need the Euclidean norms of the rows of the matrix $A(\Pi_1 A)^\dagger \in \mathbb{R}^{n \times r_1}$. Thus, we can further reduce the dimensionality of this matrix by using an ε-JLT to reduce the dimension $r_1 = \Omega(d)$ to $r_2 = O(\ln n)$. Specifically, let $\Pi_2^T \in \mathbb{R}^{r_2 \times r_1}$ be an ε-JLT for the rows of $A(\Pi_1 A)^\dagger$ (viewed as n vectors in $\mathbb{R}^{r_1}$) and consider the matrix $\Omega = A(\Pi_1 A)^\dagger \Pi_2$. This $n \times r_2$ matrix Ω may be viewed as our "randomized sketch" of the rows of $AA^\dagger$. Then, we can compute and return

$$\tilde{\ell}_i = \left\| e_i^T A (\Pi_1 A)^\dagger \Pi_2 \right\|_2^2,$$

for each $i \in [n]$, which is essentially what Algorithm 1 does. Not surprisingly, the sketch $A(\Pi_1 A)^\dagger \Pi_2$ can be used in other ways: for example, by considering the dot product between two different rows of this randomized sketching matrix (and some additional manipulations) Algorithm 2 approximates the large cross-leverage scores of A.

Input: $A \in \mathbb{R}^{n \times d}$ (with SVD $A = U\Sigma V^T$), error parameter $\varepsilon \in (0, 1/2]$.

Output: $\tilde{\ell}_i, i \in [n]$.

1. Let $\Pi_1 \in \mathbb{R}^{r_1 \times n}$ be an ε-FJLT for U, using Lemma 6 with

$$r_1 = \Omega\left(\frac{d \ln n}{\varepsilon^2} \ln\left(\frac{d \ln n}{\varepsilon^2}\right)\right).$$

2. Compute $\Pi_1 A \in \mathbb{R}^{r_1 \times d}$ and its SVD, $\Pi_1 A = U_{\Pi_1 A} \Sigma_{\Pi_1 A} V_{\Pi_1 A}^T$. Let $R^{-1} = V_{\Pi_1 A} \Sigma_{\Pi_1 A}^{-1} \in \mathbb{R}^{d \times d}$.
   (Alternatively, R could be computed by a QR factorization of $\Pi_1 A$.)

3. View the normalized rows of $AR^{-1} \in \mathbb{R}^{n \times d}$ as n vectors in $\mathbb{R}^d$, and construct $\Pi_2 \in \mathbb{R}^{d \times r_2}$ to be an ε-JLT for n^2 vectors (the aforementioned n vectors and their $n^2 - n$ pairwise sums), using Lemma 4 with

$$r_2 = O\left(\varepsilon^{-2} \ln n\right).$$

4. Construct the matrix product $\Omega = AR^{-1}\Pi_2$.

5. **For all $i \in [n]$ compute and return** $\tilde{\ell}_i = \left\|\Omega_{(i)}\right\|_2^2$.

Algorithm 1: Approximating the (diagonal) statistical leverage scores ℓ_i.

3.2 Approximating All the Statistical Leverage Scores

Our first main result is Algorithm 1, which takes as input an $n \times d$ matrix A and an error parameter $\varepsilon \in (0, 1/2]$, and returns as output numbers $\tilde{\ell}_i$, $i \in [n]$. Although the basic idea to approximate $\left\|(AA^\dagger)_{(i)}\right\|^2$ was described in the previous section, we can improve the efficiency of our approach by avoiding the full sketch of the pseudoinverse. In particular, let $\hat{A} = \Pi_1 A$ and let its SVD be $\hat{A} = U_{\hat{A}} \Sigma_{\hat{A}} V_{\hat{A}}^T$. Let $R^{-1} = V_{\hat{A}} \Sigma_{\hat{A}}^{-1}$ and note that $R^{-1} \in \mathbb{R}^{d \times d}$ is an orthogonalizer for $\hat{A}$ since $U_{\hat{A}} = \hat{A}R^{-1}$ is an orthogonal matrix.[7] In addition, note that AR^{-1} is approximately orthogonal. Thus, we can compute AR^{-1} and use it as an approximate orthogonal basis for A and then compute $\hat{\ell}_i$ as the squared row-norms of AR^{-1}. The next lemma states that this is exactly what our main algorithm does; even more, we could get the same estimates by using any "orthogonalizer" of $\Pi_1 A$.

Lemma 7 *Let R^{-1} be such that $Q = \Pi_1 A R^{-1}$ is an orthogonal matrix with* $\mathbf{rank}(Q) = \mathbf{rank}(\Pi_1 A)$. *Then,* $\left\|(AR^{-1})_{(i)}\right\|_2^2 = \hat{\ell}_i$.

---

7. This preprocessing is reminiscent of how Rokhlin and Tygert (2008) and Avron et al. (2010) preprocessed the input to provide numerical implementations of the fast relative-error algorithm (Drineas et al., 2010b) for approximate LS approximation. From this perspective, Algorithm 1 can be viewed as specifying a particular basis Q, that is, as choosing Q to be the left singular vectors of $\Pi_1 A$.

Proof Since $\hat{A} = \Pi_1 A$ has rank d (by Lemma 5) and R^{-1} preserves this rank, R^{-1} is a $d \times d$ invertible matrix. Using $\hat{A} = QR$ and properties of the pseudoinverse, we get $(\hat{A})^\dagger = R^{-1} Q^T$. Thus,

$$\hat{\ell}_i = \left\| \left(A (\Pi_1 A)^\dagger \right)_{(i)} \right\|_2^2 = \left\| \left(A R^{-1} Q^T \right)_{(i)} \right\|_2^2 = \left\| \left(A R^{-1} \right)_{(i)} Q^T \right\|_2^2 = \left\| \left(A R^{-1} \right)_{(i)} \right\|_2^2.$$

∎

This lemma says that the $\hat{\ell}_i$ of Equation (7) can be computed with any QR decomposition, rather than with the SVD; but note that one would still have to post-multiply by Π_2, as in Algorithm 1, in order to compute "quickly" the approximations of the leverage scores.

3.3 Approximating the Large Cross-leverage Scores

By combining Lemmas 9 and 10 (in Section 4.2 below) with the triangle inequality, one immediately obtains the following lemma.

Lemma 8 *Let Ω be either the sketching matrix constructed by Algorithm 1, that is, $\Omega = A R^{-1} \Pi_2$, or $\Omega = A (\Pi_1 A)^\dagger \Pi_2$ as described in Section 3.1. Then, the pairwise dot-products of the rows of Ω are additive-error approximations to the leverage scores and cross-leverage scores:*

$$\left| \langle U_{(i)}, U_{(j)} \rangle - \langle \Omega_{(i)}, \Omega_{(j)} \rangle \right| \leq \frac{3\varepsilon}{1 - \varepsilon} \| U_{(i)} \|_2 \| U_{(j)} \|_2.$$

That is, if one were interested in obtaining an approximation to all the cross-leverage scores to within additive error (and thus the diagonal statistical leverage scores to relative-error), then the algorithm which first computes Ω followed by all the pairwise inner products achieves this in time $T(\Omega) + O\left(n^2 r_2 \right)$, where $T(\Omega)$ is the time to compute Ω from Section 3.2 and $r_2 = O(\varepsilon^{-2} \ln n)$.[8] The challenge is to avoid the n^2 computational complexity and this can be done if one is interested only in the large cross-leverage scores.

   Our second main result is provided by Algorithms 2 and 3. Algorithm 2 takes as input an $n \times d$ matrix A, a parameter $\kappa > 1$, and an error parameter $\varepsilon \in (0, 1/2]$, and returns as output a subset of $[n] \times [n]$ and estimates $\tilde{c}_{ij}$ satisfying Theorem 3. The first step of the algorithm is to compute the matrix $\Omega = A R^{-1} \Pi_2$ constructed by Algorithm 1. Then, Algorithm 2 uses Algorithm 3 as a subroutine to compute "heavy hitter" pairs of rows from a matrix.

4. Proofs of Our Main Theorems

We will start with a sketch of the proofs of Theorems 2 and 3, and then we will provide the details.

4.1 Sketch of the Proof of Theorems 2 and 3

We will start by providing a sketch of the proof of Theorems 2 and 3. A detailed proof is provided in the next two subsections. In our analysis, we will condition on the events that $\Pi_1 \in \mathbb{R}^{r_1 \times n}$ is an ε-FJLT for U and $\Pi_2 \in \mathbb{R}^{r_1 \times r_2}$ is an ε-JLT for n^2 points in $\mathbb{R}^{r_1}$. Note that by setting $\delta = 0.1$ in Lemma 4, both events hold with probability at least 0.8, which is equal to the success probability of Theorems 2 and 3. The algorithm estimates $\tilde{\ell}_i = \|\tilde{u}_i\|_2^2$, where $\tilde{u}_i = e_i^T A (\Pi_1 A)^\dagger \Pi_2$. First, observe

---

8. The exact algorithm which computes a basis first and then the pairwise inner products requires $O(nd^2 + n^2 d)$ time. Thus, by using the sketch, we can already improve on this running time by a factor of $d / \ln n$.

Input: $A \in \mathbb{R}^{n \times d}$ and parameters $\kappa > 1$, $\varepsilon \in (0, 1/2]$.

Output: The set $\mathcal{H}$ consisting of pairs (i, j) together with estimates $\tilde{c}_{ij}$ satisfying Theorem 3.

1. Compute the $n \times r_2$ matrix $\Omega = AR^{-1}\Pi_2$ from Algorithm 1.

2. Use Algorithm 3 with inputs Ω and $\kappa' = \kappa(1 + 30d\varepsilon)$ to obtain the set $\mathcal{H}$ containing all the κ'-heavy pairs of Ω.

3. **Return** the pairs in $\mathcal{H}$ as the κ-heavy pairs of A.

Algorithm 2: Approximating the large (off-diagonal) cross-leverage scores c_{ij}.

Input: $X \in \mathbb{R}^{n \times r}$ with rows $x_1, \ldots, x_n$ and a parameter $\kappa > 1$.

Output: $\mathcal{H} = \{(i, j), \tilde{c}_{ij}\}$ containing all heavy (unordered) pairs. The pair $(i, j), \tilde{c}_{ij} \in \mathcal{H}$ if and only if $\tilde{c}_{ij}^2 = \langle x_i, x_j \rangle^2 \geq \left\| X^T X \right\|_F^2 / \kappa$.

1: Compute the norms $\|x_i\|_2$ and sort the rows according to norm, so that
$\|x_1\|_2 \leq \cdots \leq \|x_n\|_2$.
2: $\mathcal{H} \leftarrow \{\}$; $z_1 \leftarrow n$; $z_2 \leftarrow 1$.
3: **while** $z_2 \leq z_1$ **do**
4:     **while** $\|x_{z_1}\|_2^2 \|x_{z_2}\|_2^2 < \left\| X^T X \right\|_F^2 / \kappa$ **do**
5:         $z_2 \leftarrow z_2 + 1$.
6:         **if** $z_2 > z_1$ **then**
7:             **return** $\mathcal{H}$.
8:         **end if**
9:     **end while**
10:     **for each** pair (i, j) where $i = z_1$ and $j \in \{z_2, z_2 + 1, \ldots, z_1\}$ **do**
11:         $\tilde{c}_{ij}^2 = \langle x_i, x_j \rangle^2$.
12:         **if** $\tilde{c}_{ij}^2 \geq \left\| X^T X \right\|_F^2 / \kappa$ **then**
13:             add (i, j) and $\tilde{c}_{ij}$ to $\mathcal{H}$.
14:         **end if**
15:         $z_1 \leftarrow z_1 - 1$.
16:     **end for**
17: **end while**
18: **return** $\mathcal{H}$.

Algorithm 3: Computing heavy pairs of a matrix.

that the sole purpose of Π_2 is to improve the running time while preserving pairwise inner products;

this is achieved because Π_2 is an ε-JLT for n^2 points. So, the results will follow if

$$e_i^T A (\Pi_1 A)^\dagger ((\Pi_1 A)^\dagger)^T A^T e_j \approx e_i^T U U^T e_j$$

and $(\Pi_1 A)^\dagger$ can be computed efficiently. Since Π_1 is an ε-FJLT for U, where $A = U\Sigma V^T$, $(\Pi_1 A)^\dagger$ can be computed in $O(nd \ln r_1 + r_1 d^2)$ time. By Lemma 5, $(\Pi_1 A)^\dagger = V\Sigma^{-1}(\Pi_1 U)^\dagger$, and so

$$e_i^T A (\Pi_1 A)^\dagger ((\Pi_1 A)^\dagger)^T A^T e_j = e_i^T U (\Pi_1 U)^\dagger (\Pi_1 U)^{\dagger^T} U^T e_j.$$

Since Π_1 is an ε-FJLT for U, it follows that $(\Pi_1 U)^\dagger (\Pi_1 U)^{\dagger^T} \approx I_d$, that is, that $\Pi_1 U$ is approximately orthogonal. Theorem 2 follows from this basic idea. However, in order to prove Theorem 3, having a sketch which preserves inner products alone is not sufficient. We also need a fast algorithm to identify the large inner products and to relate these to the actual cross-leverage scores. Indeed, it is possible to efficiently find pairs of rows in a general matrix with large inner products. Combining this with the fact that the inner products are preserved, we obtain Theorem 3.

4.2 Proof of Theorem 2

We condition all our analysis on the events that $\Pi_1 \in \mathbb{R}^{r_1 \times n}$ is an ε-FJLT for U and $\Pi_2 \in \mathbb{R}^{r_1 \times r_2}$ is an ε-JLT for n^2 points in $\mathbb{R}^{r_1}$. Define

$$\begin{aligned}
\hat{u}_i &= e_i^T A (\Pi_1 A)^\dagger, \text{ and} \\
\tilde{u}_i &= e_i^T A (\Pi_1 A)^\dagger \Pi_2.
\end{aligned}$$

Then, $\hat{\ell}_i = \|\hat{u}_i\|_2^2$ and $\tilde{\ell}_i = \|\tilde{u}_i\|_2^2$. The proof will follow from the following two lemmas.

Lemma 9 *For $i, j \in [n]$,*

$$\left| \langle U_{(i)}, U_{(j)} \rangle - \langle \hat{u}_i, \hat{u}_j \rangle \right| \leq \frac{\varepsilon}{1-\varepsilon} \left\| U_{(i)} \right\|_2 \left\| U_{(j)} \right\|_2.$$

Lemma 10 *For $i, j \in [n]$,*

$$\left| \langle \hat{u}_i, \hat{u}_j \rangle - \langle \tilde{u}_i, \tilde{u}_j \rangle \right| \leq 2\varepsilon \|\hat{u}_i\|_2 \left\| \hat{u}_j \right\|_2.$$

Lemma 9 states that $\langle \hat{u}_i, \hat{u}_j \rangle$ is an additive error approximation to all the cross-leverage scores ($i \neq j$) and a relative error approximation for the diagonals ($i = j$). Similarly, Lemma 10 shows that these cross-leverage scores are preserved by Π_2. Indeed, with $i = j$, from Lemma 9 we have $|\hat{\ell}_i - \ell_i| \leq \frac{\varepsilon}{1-\varepsilon}\ell_i$, and from Lemma 10 we have $|\hat{\ell}_i - \tilde{\ell}_i| \leq 2\varepsilon\hat{\ell}_i$. Using the triangle inequality and $\varepsilon \leq 1/2$:

$$\left| \ell_i - \tilde{\ell}_i \right| = \left| \ell_i - \hat{\ell}_i + \hat{\ell}_i - \tilde{\ell}_i \right| \leq \left| \ell_i - \hat{\ell}_i \right| + \left| \hat{\ell}_i - \tilde{\ell}_i \right| \leq \left(\frac{\varepsilon}{1-\varepsilon} + 2\varepsilon \right) \ell_i \leq 4\varepsilon\ell_i.$$

The theorem follows after rescaling ε.

Proof of Lemma 9 Let $A = U\Sigma V^T$. Using this SVD of A and Equation (4) in Lemma 5,

$$\langle \hat{u}_i, \hat{u}_j \rangle \;=\; e_i^T U\Sigma V^T V\Sigma^{-1} (\Pi_1 U)^\dagger (\Pi_1 U)^{\dagger^T} \Sigma^{-1} V^T V\Sigma U^T e_j = e_i^T U (\Pi_1 U)^\dagger (\Pi_1 U)^{\dagger^T} U^T e_j.$$

By performing standard manipulations, we can now bound $\left| \langle U_{(i)}, U_{(j)} \rangle - \langle \hat{u}_i, \hat{u}_j \rangle \right|$:

$$\begin{aligned}
\left| \langle U_{(i)}, U_{(j)} \rangle - \langle \hat{u}_i, \hat{u}_j \rangle \right| &= e_i^T UU^T e_j - e_i^T U (\Pi_1 U)^\dagger (\Pi_1 U)^{\dagger T} U^T e_j \\
&= e_i^T U \left(I_d - (\Pi_1 U)^\dagger (\Pi_1 U)^{\dagger T} \right) U^T e_j \\
&\leq \left\| I_d - (\Pi_1 U)^\dagger (\Pi_1 U)^{\dagger T} \right\|_2 \left\| U_{(i)} \right\|_2 \left\| U_{(j)} \right\|_2.
\end{aligned}$$

Let the SVD of $\Psi = \Pi_1 U$ be $\Psi = U_\Psi \Sigma_\Psi V_\Psi^T$, where V_Ψ is a full rotation in d dimensions (because $\mathbf{rank}(A) = \mathbf{rank}(\Pi_1 U)$). Then, $\Psi^\dagger \Psi^{\dagger^T} = V_\Psi \Sigma_\Psi^{-2} V_\Psi^T$. Thus,

$$\begin{aligned}
\left| \langle U_{(i)}, U_{(j)} \rangle - \langle \hat{u}_i, \hat{u}_j \rangle \right| &\leq \left\| I_d - V_\Psi \Sigma_\Psi^{-2} V_\Psi^T \right\|_2 \left\| U_{(i)} \right\|_2 \left\| U_{(j)} \right\|_2 \\
&= \left\| V_\Psi V_\Psi^T - V_\Psi \Sigma_\Psi^{-2} V_\Psi^T \right\|_2 \left\| U_{(i)} \right\|_2 \left\| U_{(j)} \right\|_2 \\
&= \left\| I_d - \Sigma_\Psi^{-2} \right\|_2 \left\| U_{(i)} \right\|_2 \left\| U_{(j)} \right\|_2,
\end{aligned}$$

where we used the fact that $V_\Psi V_\Psi^T = V_\Psi^T V_\Psi = I_d$ and the unitary invariance of the spectral norm. Finally, using Equation (3) of Lemma 5 the result follows.

Proof of Lemma 10. Since Π_2 is an ε-JLT for n^2 vectors, it preserves the norms of an arbitrary (but fixed) collection of n^2 vectors. Let $x_i = \hat{u}_i / \|\hat{u}_i\|_2$. Consider the following n^2 vectors:

$$\begin{aligned}
x_i \qquad &\text{for } i \in [n], \text{ and} \\
x_i + x_j \quad &\text{for } i, j \in [n], i \neq j.
\end{aligned}$$

By the ε-JLT property of Π_2 and the fact that $\|x_i\|_2 = 1$,

$$1 - \varepsilon \leq \|x_i \Pi_2\|_2^2 \leq 1 + \varepsilon \qquad\qquad \text{for } i \in [n], \text{ and} \qquad (8)$$
$$(1-\varepsilon)\|x_i + x_j\|_2^2 \leq \|x_i \Pi_2 + x_j \Pi_2\|_2^2 \leq (1+\varepsilon)\|x_i + x_j\|_2^2 \quad \text{for } i, j \in [n], i \neq j. \qquad (9)$$

Combining Equations (8) and (9) after expanding the squares using the identity $\|a+b\|^2 = \|a\|^2 + \|b\|^2 + 2\langle a, b \rangle$, substituting $\|x_i\| = 1$, and after some algebra, we obtain

$$\langle x_i, x_j \rangle - 2\varepsilon \leq \langle x_i \Pi_2, x_j \Pi_2 \rangle \leq \langle x_i, x_j \rangle + 2\varepsilon.$$

To conclude the proof, multiply throughout by $\|\hat{u}_i\| \|\hat{u}_j\|$ and use the homogeneity of the inner product, together with the linearity of Π_2, to obtain:

$$\langle \hat{u}_i, \hat{u}_j \rangle - 2\varepsilon \|\hat{u}_i\| \|\hat{u}_j\| \leq \langle \hat{u}_i \Pi_2, \hat{u}_j \Pi_2 \rangle \leq \langle \hat{u}_i, \hat{u}_j \rangle + 2\varepsilon \|\hat{u}_i\| \|\hat{u}_j\|.$$

Running Times. By Lemma 7, we can use $V_{\Pi_1 A} \Sigma_{\Pi_1 A}^{-1}$ instead of $(\Pi_1 A)^\dagger$ and obtain the same estimates. Since Π_1 is an ε-FJLT, the product $\Pi_1 A$ can be computed in $O(nd \ln r_1)$ while its SVD takes an additional $O(r_1 d^2)$ time to return $V_{\Pi_1 A} \Sigma_{\Pi_1 A}^{-1} \in \mathbb{R}^{d \times d}$. Since $\Pi_2 \in \mathbb{R}^{d \times r_2}$, we obtain $V_{\Pi_1 A} \Sigma_{\Pi_1 A}^{-1} \Pi_2 \in \mathbb{R}^{d \times r_2}$ in an additional $O(r_2 d^2)$ time. Finally, premultiplying by A takes $O(ndr_2)$

time, and computing and returning the squared row-norms of $\Omega = AV_{\Pi_1 A}\Sigma_{\Pi_1 A}^{-1}\Pi_2 \in \mathbb{R}^{n \times r_2}$ takes $O(nr_2)$ time. So, the total running time is the sum of all these operations, which is

$$O(nd \ln r_1 + ndr_2 + r_1 d^2 + r_2 d^2).$$

Recall that for our implementations of the ε-JLTs and ε-FJLTs, we have $\delta = 0.1$, and we have $r_1 = O\left(\varepsilon^{-2}d\left(\ln n\right)\left(\ln\left(\varepsilon^{-2}d\ln n\right)\right)\right)$ and $r_2 = O(\varepsilon^{-2}\ln n)$. It follows that the asymptotic running time is

$$O\left(nd \ln\left(d\varepsilon^{-1}\right) + nd\varepsilon^{-2}\ln n + d^3\varepsilon^{-2}\left(\ln n\right)\left(\ln\left(d\varepsilon^{-1}\right)\right)\right).$$

To simplify, suppose that $d \leq n \leq e^d$ and treat ε as a constant. Then, the asymptotic running time is

$$O\left(nd \ln n + d^3\left(\ln n\right)\left(\ln d\right)\right).$$

4.3 Proof of Theorem 3

We first construct an algorithm to estimate the large inner products among the rows of an arbitrary matrix $X \in \mathbb{R}^{n \times r}$ with $n > r$. This general algorithm will be applied to the matrix $\Omega = AV_{\Pi_1 A}\Sigma_{\Pi_1 A}^{-1}\Pi_2$. Let $x_1, \ldots, x_n$ denote the rows of X; for a given $\kappa > 1$, the pair (i, j) is *heavy* if

$$\langle x_i, x_j \rangle^2 \geq \frac{1}{\kappa}\left\|X^T X\right\|_F^2.$$

By the Cauchy-Schwarz inequality, this implies that

$$\|x_i\|_2^2 \|x_j\|_2^2 \geq \frac{1}{\kappa}\left\|X^T X\right\|_F^2, \tag{10}$$

so it suffices to find all the pairs (i, j) for which Equation (10) holds. We will call such pairs *norm-heavy*. Let s be the number of norm-heavy pairs satisfying Equation (10). We first bound the number of such pairs.

Lemma 11 *Using the above notation, $s \leq \kappa r$.*

Proof Observe that

$$\sum_{i,j=1}^{n} \|x_i\|_2^2 \|x_j\|_2^2 = \left(\sum_{i=1}^{n} \|x_i\|_2^2\right)^2 = \|X\|_F^4 = \left(\sum_{i=1}^{r} \sigma_i^2\right)^2,$$

where $\sigma_1, \ldots, \sigma_r$ are the singular values of X. To conclude, by the definition of a heavy pair,

$$\sum_{i,j} \|x_i\|_2^2 \|x_j\|_2^2 \geq \frac{s}{\kappa}\left\|X^T X\right\|_F^2 = \frac{s}{\kappa}\sum_{i=1}^{r} \sigma_i^4 \geq \frac{s}{\kappa r}\left(\sum_{i=1}^{r} \sigma_i^2\right)^2,$$

where the last inequality follows by Cauchy-Schwarz. ∎

Algorithm 3 starts by computing the norms $\|x_i\|_2^2$ for all $i \in [n]$ and sorting them (in $O(nr + n\ln n)$ time) so that we can assume that $\|x_1\|_2 \leq \cdots \leq \|x_n\|_2$. Then, we initialize the set of norm-heavy pairs to $\mathcal{H} = \{\}$ and we also initialize two pointers $z_1 = n$ and $z_2 = 1$. The basic loop in the algorithm checks if $z_2 > z_1$ and stops if that is the case. Otherwise, we increment z_2 to the first pair (z_1, z_2) that

is norm-heavy. If none of pairs are norm heavy (i.e., $z_2 > z_1$ occurs), then we stop and output $\mathcal{H}$; otherwise, we add $(z_1, z_2), (z_1, z_2 + 1), \ldots, (z_1, z_1)$ to $\mathcal{H}$. This basic loop computes all pairs (z_1, i) with $i \le z_1$ that are norm-heavy. Next, we decrease z_1 by one and if $z_1 < z_2$ we stop and output $\mathcal{H}$; otherwise, we repeat the basic loop. Note that in the basic loop z_2 is always *incremented*. This occurs whenever the pair (z_1, z_2) is not norm-heavy. Since z_2 can be incremented at most n times, the number of times we check whether a pair is norm-heavy and fail is at most n. Every successful check results in the addition of at least one norm-heavy pair into $\mathcal{H}$ and thus the number of times we check if a pair is norm heavy (a constant-time operation) is at most $n + s$. The number of pair additions into $\mathcal{H}$ is exactly s and thus the total running time is $O(nr + n \ln n + s)$. Finally, we must check each norm-heavy pair to verify whether or not it is actually heavy by computing s inner products vectors in $\mathbb{R}^r$; this can be done in $O(sr)$ time. Using $s \le \kappa r$ we get the following lemma.

Lemma 12 *Algorithm 3 returns $\mathcal{H}$ including all the heavy pairs of X in $O(nr + \kappa r^2 + n \ln n)$ time.*

To complete the proof, we apply Algorithm 3 with $\Omega = A V_{\Pi_1 A} \Sigma_{\Pi_1 A}^{-1} \Pi_2 \in \mathbb{R}^{n \times r_2}$, where $r_2 = O(\varepsilon^{-2} \ln n)$. Let $\tilde{u}_1, \ldots, \tilde{u}_n$ denote the rows of Ω and recall that $A = U \Sigma V^T$. Let $u_1, \ldots, u_n$ denote the rows of U; then, from Lemma 8,

$$\langle u_i, u_j \rangle - \frac{3\varepsilon}{1 - \varepsilon} \|u_i\| \|u_j\| \le \langle \tilde{u}_i, \tilde{u}_j \rangle \le \langle u_i, u_j \rangle + \frac{3\varepsilon}{1 - \varepsilon} \|u_i\| \|u_j\|. \tag{11}$$

Given ε, κ, assume that for the pair of vectors u_i and u_j

$$\langle u_i, u_j \rangle^2 \ge \frac{1}{\kappa} \|U^T U\|_F^2 + 12\varepsilon \|u_i\|^2 \|u_j\|^2 = \frac{d}{\kappa} + 12\varepsilon \|u_i\|^2 \|u_j\|^2,$$

where the last equality follows from $\|U^T U\|_F^2 = \|I_d\|_F^2 = d$. By Equation (11), after squaring and using $\varepsilon < 0.5$,

$$\langle u_i, u_j \rangle^2 - 12\varepsilon \|u_i\|^2 \varepsilon \|u_j\|^2 \le \langle \tilde{u}_i, \tilde{u}_j \rangle^2 \le \langle u_i, u_j \rangle^2 + 30\varepsilon \|u_i\|^2 \|u_j\|^2. \tag{12}$$

Thus, $\langle \tilde{u}_i, \tilde{u}_j \rangle^2 \ge d/\kappa$ and summing Equation (12) over all i, j we get $\|\Omega^T \Omega\|_F^2 \le d + 30\varepsilon d^2$, or, equivalently,

$$d \ge \frac{\|\Omega^T \Omega\|_F^2}{1 + 30 d\varepsilon}.$$

We conclude that

$$\langle u_i, u_j \rangle^2 \ge \frac{d}{\kappa} + 12\varepsilon \|u_i\|^2 \|u_j\|^2 \implies \langle \tilde{u}_i, \tilde{u}_j \rangle^2 \ge \frac{d}{\kappa} \ge \frac{\|\Omega^T \Omega\|_F^2}{\kappa(1 + 30 d\varepsilon)}. \tag{13}$$

By construction, Algorithm 3 is invoked with $\kappa' = \kappa \|\Omega^T \Omega\|_F^2 / d$ and thus it finds all pairs with $\langle \tilde{u}_i, \tilde{u}_j \rangle^2 \ge \|\Omega^T \Omega\|_F^2 / \kappa' = d/\kappa$. This set contains all pairs for which

$$\langle u_i, u_j \rangle^2 \ge \frac{d}{\kappa} + 12\varepsilon \|u_i\|^2 \|u_j\|^2.$$

Further, since every pair returned satisfies $\langle \tilde{u}_i, \tilde{u}_j \rangle^2 \ge d/\kappa$, by Equation (12), $c_{ij} \ge d/\kappa - 30\varepsilon \ell_i \ell_j$. This proves the first claim of the Theorem; the second claim follows analogously from Equation (12).

Using Lemma 12, the running time of our approach is $O(nr_2 + \kappa' r_2^2 + n \ln n)$. Since $r_2 = O(\varepsilon^{-2} \ln n)$, and, by Equation (13), $\kappa' = \kappa \|\Omega^T \Omega\|_F^2 / d \le \kappa(1 + 30 d\varepsilon)$, the overall running time is $O(\varepsilon^{-2} n \ln n + \varepsilon^{-3} \kappa d \ln^2 n)$.

5. Extending Our Algorithm to General Matrices

In this section, we will describe an important extension of our main result, namely the computation of the statistical leverage scores relative to the best rank-k approximation to a general matrix A. More specifically, we consider the estimation of leverage scores for the case of general "fat" matrices, namely input matrices $A \in \mathbb{R}^{n \times d}$, where both n and d are large, for example, when $d = n$ or $d = \Theta(n)$. Clearly, the leverage scores of any full rank $n \times n$ matrix are exactly uniform. The problem becomes interesting if one specifies a rank parameter $k \ll \min\{n, d\}$. This may arise when the numerical rank of A is small (e.g., in some scientific computing applications, more than 99% of the spectral norm of A may be captured by some $k \ll \min\{n, d\}$ directions), or, more generally, when one is interested in some low rank approximation to A (e.g., in some data analysis applications, a reasonable fraction or even the majority of the Frobenius norm of A may be captured by some $k \ll \min\{n, d\}$ directions, where k is determined by some exogenously-specified model selection criterion). Thus, assume that in addition to a general $n \times d$ matrix A, a rank parameter $k < \min\{n, d\}$ is specified. In this case, we wish to obtain the statistical leverage scores $\ell_i = \left\| (U_k)_{(i)} \right\|_2^2$ for $A_k = U_k \Sigma_k V_k^T$, the best rank-k approximation to A. Equivalently, we seek the normalized leverage scores

$$p_i = \frac{\ell_i}{k}. \tag{14}$$

Note that $\sum_{i=1}^n p_i = 1$ since $\sum_{i=1}^n \ell_i = \|U_k\|_F^2 = k$.

Unfortunately, as stated, this is an ill-posed problem. Indeed, consider the degenerate case when $A = I_n$ (i.e., the $n \times n$ identity matrix). In this case, U_k is not unique and the leverage scores are not well-defined. Moreover, for the obvious $\binom{n}{k}$ equivalent choices for U_k, the leverage scores defined according to any one of these choices do not provide a relative error approximation to the leverage scores defined according to any other choices. More generally, removing this trivial degeneracy does not help. Consider the matrix

$$A = \begin{pmatrix} I_k & 0 \\ 0 & (1-\gamma)I_{n-k} \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

In this example, the leverage scores for A_k are well defined. However, as $\gamma \to 0$, it is not possible to distinguish between the top-k singular space and its complement. This example suggests that it should be possible to obtain some result conditioning on the spectral gap at the kth singular value. For example, one might assume that $\sigma_k^2 - \sigma_{k+1}^2 \geq \gamma > 0$, in which case the parameter γ would play an important role in the ability to solve this problem. Any algorithm which cannot distinguish the singular values with an error less than γ will confuse the k-th and $(k+1)$-th singular vectors and consequently will fail to get an accurate approximation to the leverage scores for A_k.

In the following, we take a more natural approach which leads to a clean problem formulation. To do so, recall that the leverage scores and the related normalized leverage scores of Equation (14) are used to approximate the matrix in some way, for example, we might be seeking a low-rank approximation to the matrix with respect to the spectral (Drineas et al., 2008) or the Frobenius (Boutsidis et al., 2009) norm, or we might be seeking useful features or data points in downstream data analysis applications (Paschou et al., 2007; Mahoney and Drineas, 2009), or we might be seeking to develop high-quality numerical implementations of low-rank matrix approximation algorithms (Halko et al., 2011), etc. In all these cases, we only care that the estimated leverage scores are a good approximation to the leverage scores of some "good" low-rank approximation to A. The

following definition captures the notion of a set of rank-k matrices that are good approximations to A.

Definition 13 *Given $A \in \mathbb{R}^{n \times d}$ and a rank parameter $k \ll \min\{n, d\}$, let A_k be the best rank-k approximation to A. Define the set $\mathcal{S}_\varepsilon$ of rank-k matrices that are good approximations to A as follows (for $\xi = 2, F$):*

$$\mathcal{S}_\varepsilon = \left\{ X \in \mathbb{R}^{n \times d} : \mathbf{rank}(X) = k \ \text{ and } \ \|A - X\|_\xi \leq (1 + \varepsilon)\|A - A_k\|_\xi \right\}.$$

We are now ready to define our approximations to the normalized leverage scores of any matrix $A \in \mathbb{R}^{n \times d}$ given a rank parameter $k \ll \min\{n, d\}$. Instead of seeking to approximate the p_i of Equation (14) (a problem that is ill-posed as discussed above), we will be satisfied if we can approximate the normalized leverage scores of some matrix $X \in \mathcal{S}_\varepsilon$. This is an interesting relaxation of the task at hand: all matrices X that are sufficiently close to A_k are essentially equivalent, since they can be used instead of A_k in applications.

Definition 14 *Given $A \in \mathbb{R}^{n \times d}$ and a rank parameter $k \ll \min\{n, d\}$, let $\mathcal{S}_\varepsilon$ be the set of matrices of Definition 13. We call the numbers $\hat{p}_i$ (for all $i \in [n]$) β-approximations to the normalized leverage scores of A_k (the best rank-k approximation to A) if, for some matrix $X \in \mathcal{S}_\varepsilon$,*

$$\hat{p}_i \geq \frac{\beta \|(U_X)_{(i)}\|_2^2}{k} \qquad \text{and} \qquad \sum_{i=1}^{n} \hat{p}_i = 1.$$

Here $U_X \in \mathbb{R}^{n \times k}$ is the matrix of the left singular vectors of X.

Thus, we will seek algorithms whose output is a set of numbers, with the requirement that those numbers are good approximations to the normalized leverage scores of some matrix $X \in \mathcal{S}_\varepsilon$ (instead of A_k). This removes the ill-posedness of the original problem. Next, we will give two examples of algorithms that compute such β-approximations to the normalized leverage scores of a general matrix A with a rank parameter k for two popular norms, the spectral norm and the Frobenius norm.[9]

5.1 Leverage Scores for Spectral Norm Approximators

Algorithm 4 approximates the statistical leverage scores of a general matrix A with rank parameter k in the spectral norm case. It takes as inputs a matrix $A \in \mathbb{R}^{n \times d}$ with $\mathbf{rank}(A) = \rho$ and a rank parameter $k \ll \rho$, and outputs a set of numbers $\hat{p}_i$ for all $i \in [n]$, namely our approximations to the normalized leverage scores of A with rank parameter k.

The next lemma argues that there exists a matrix $X \in \mathbb{R}^{n \times d}$ of rank k that is sufficiently close to A (in particular, it is a member of $\mathcal{S}_\varepsilon$ with constant probability) and, additionally, can be written as $X = BY$, where $Y \in \mathbb{R}^{2k \times d}$ is a matrix of rank k. A version of this lemma was essentially proven in Halko et al. (2011), but see also Rokhlin et al. (2009) for computational details; we will use the version of the lemma that appeared in Boutsidis et al. (2011b). (See also the conference version (Boutsidis et al., 2011a), but in the remainder we refer to the technical report version (Boutsidis et al., 2011b) for consistency of numbering.) Note that for our purposes in this section, the computation of Y is not relevant and we defer the reader to Halko et al. (2011) and Boutsidis et al. (2011b) for details.

---

9. Note that we will not compute $\mathcal{S}_\varepsilon$, but our algorithms will compute a matrix in that set. Moreover, that matrix can be used for high-quality low-rank matrix approximation. See the comments in Section 1.4 for more details.

Input: $A \in \mathbb{R}^{n \times d}$ with **rank**$(A) = \rho$ and a rank parameter $k \ll \rho$

Output: $\hat{p}_i, i \in [n]$

1. Construct $\Pi \in \mathbb{R}^{d \times 2k}$ with entries drawn in i.i.d. trials from the normal distribution $\mathcal{N}(0,1)$.

2. Compute $B = \left(AA^T\right)^q A\Pi \in \mathbb{R}^{n \times 2k}$, with q as in Equation (16).

3. Approximately compute the statistical leverage scores of the "tall" matrix B by calling Algorithm 1 with inputs B and ε; let $\hat{\ell}_i$ (for all $i \in [n]$) be the outputs of Algorithm 1.

4. Return

$$\hat{p}_i = \frac{\hat{\ell}_i}{\sum_{j=1}^n \hat{\ell}_j} \tag{15}$$

for all $i \in [n]$.

Algorithm 4: Approximating the statistical leverage scores of a general matrix A (spectral norm case).

Lemma 15 (Spectral Sketch) *Given $A \in \mathbb{R}^{n \times d}$ of rank ρ, a rank parameter k such that $2 \le k < \rho$, and an error parameter ε such that $0 < \varepsilon < 1$, let $\Pi \in \mathbb{R}^{d \times 2k}$ be a standard Gaussian matrix (with entries selected in i.i.d. trials from $\mathcal{N}(0,1)$). If $B = \left(AA^T\right)^q A\Pi$, where*

$$q \ge \left\lceil \frac{\ln\left(1 + \sqrt{\frac{k}{k-1}} + e\sqrt{\frac{2}{k}}\sqrt{\min\{n,d\} - k}\right)}{2\ln(1 + \varepsilon/10) - 1/2} \right\rceil, \tag{16}$$

then there exists a matrix $X \in \mathbb{R}^{n \times d}$ of rank k satisfying $X = BY$ (with $Y \in \mathbb{R}^{2k \times d}$) such that

$$\mathbf{E}\left[\|A - X\|_2\right] \le \left(1 + \frac{\varepsilon}{10}\right)\|A - A_k\|_2.$$

The matrix B can be computed in $O(ndkq)$ time.

This version of the above lemma is proven in Boutsidis et al. (2011b).[10] Now, since X has rank k, it follows that $\|A - X\|_2 \ge \|A - A_k\|_2$ and thus we can consider the non-negative random variable $\|A - X\|_2 - \|A - A_k\|_2$ and apply Markov's inequality to get that

$$\|A - X\|_2 - \|A - A_k\|_2 \le \varepsilon\|A - A_k\|_2$$

holds with probability at least 0.9. Thus, $X \in \mathcal{S}_\varepsilon$ with probability at least 0.9.

---

10. More specifically, the proof may be found in Lemma 32 and in particular in Equation (14) in Section A.2; note that for our purposes here we replaced $\varepsilon/\sqrt{2}$ by $\varepsilon/10$ after adjusting q accordingly.

The next step of the proposed algorithm is to approximately compute the leverage scores of $B \in \mathbb{R}^{n \times 2k}$ via Algorithm 1. Under the assumptions of Theorem 2, this step runs in $O\left(nk\varepsilon^{-2}\ln n\right)$ time. Let $U_X \in \mathbb{R}^{n \times k}$ be the matrix containing the left singular vectors of the matrix X of Lemma 15. Then, since $X = BY$ by Lemma 15, it follows that

$$U_B = [U_X \ U_R]$$

is a basis for the subspace spanned by the columns of B. Here $U_R \in \mathbb{R}^{n \times k}$ is an orthogonal matrix whose columns are perpendicular to the columns of U_X. Now consider the approximate leverage scores $\hat{\ell}_i$ computed by Algorithm 1 and note that (by Theorem 2),

$$\left| \hat{\ell}_i - \left\| (U_B)_{(i)} \right\|_2^2 \right| \leq \varepsilon \left\| (U_B)_{(i)} \right\|_2^2$$

holds with probability at least 0.8 for all $i \in [n]$. It follows that

$$\sum_{j=1}^n \hat{\ell}_j \leq (1+\varepsilon) \sum_{j=1}^n \left\| (U_B)_{(j)} \right\|_2^2 = (1+\varepsilon) \sum_{j=1}^n \|U_B\|_F^2 = 2(1+\varepsilon)k.$$

Finally,

$$
\begin{aligned}
\hat{p}_i = \frac{\hat{\ell}_i}{\sum_{j=1}^n \hat{\ell}_j} \quad &\geq \quad (1-\varepsilon)\frac{\left\| (U_B)_{(i)} \right\|_2^2}{\sum_{j=1}^n \hat{\ell}_j} \\
&\geq \quad (1-\varepsilon)\frac{\left\| (U_X)_{(i)} \right\|_2^2 + \left\| (U_R)_{(i)} \right\|_2^2}{\sum_{j=1}^n \hat{\ell}_j} \\
&\geq \quad \frac{1-\varepsilon}{2}\frac{\left\| (U_X)_{(i)} \right\|_2^2}{\sum_{j=1}^n \hat{\ell}_j} \\
&\geq \quad \frac{1-\varepsilon}{2(1+\varepsilon)}\frac{\left\| (U_X)_{(i)} \right\|_2^2}{k}.
\end{aligned}
$$

Clearly, $\left\| (U_X)_{(i)} \right\|_2^2 / k$ are the normalized leverage scores of the matrix X. Recall that $X \in \mathcal{S}_\varepsilon$ with probability at least 0.9 and use Definition 14 to conclude that the scores $\hat{p}_i$ of Equation (15) are $\left(\frac{1-\varepsilon}{2(1+\varepsilon)} \right)$-approximations to the normalized leverage scores of A with rank parameter k. The following Theorem summarizes the above discussion:

Theorem 16 *Given $A \in \mathbb{R}^{n \times d}$, a rank parameter k, and an accuracy parameter ε, Algorithm 4 computes a set of normalized leverage scores $\hat{p}_i$ that are $\left(\frac{1-\varepsilon}{2(1+\varepsilon)} \right)$-approximations to the normalized leverage scores of A with rank parameter k with probability at least 0.7. The proposed algorithm runs in*

$$O\left(ndk\frac{\ln(\min\{n,d\})}{\ln(1+\varepsilon)} + nk\varepsilon^{-2}\ln n \right)$$

time.

Input: $A \in \mathbb{R}^{n \times d}$ with $\mathbf{rank}(A) = \rho$ and a rank parameter $k \ll \rho$

Output: $\hat{p}_i, i \in [n]$

1. Let r be as in Equation (18) and construct $\Pi \in \mathbb{R}^{d \times r}$ whose entries are drawn in i.i.d. trials from the normal distribution $\mathcal{N}(0,1)$.

2. Compute $B = A\Pi \in \mathbb{R}^{n \times r}$.

3. Compute a matrix $Q \in \mathbb{R}^{n \times r}$ whose columns form an orthonormal basis for the column space of B.

4. Compute the matrix $Q^T A \in \mathbb{R}^{r \times d}$ and its left singular vectors $U_{Q^T A} \in \mathbb{R}^{r \times d}$.

5. Let $U_{Q^T A, k} \in \mathbb{R}^{r \times k}$ denote the top k left singular vectors of the matrix $Q^T A$ (the first k columns of $U_{Q^T A}$) and compute, for all $i \in [n]$,

$$\hat{\ell}_i = \left\| \left(Q U_{Q^T A, k} \right)_{(i)} \right\|_2^2. \tag{17}$$

6. Return $\hat{p}_i = \hat{\ell}_i / k$ for all $i \in [n]$.

Algorithm 5: Approximating the statistical leverage scores of a general matrix A (Frobenius norm case).

5.2 Leverage Scores for Frobenius Norm Approximators

Algorithm 5 approximates the statistical leverage scores of a general matrix A with rank parameter k in the Frobenius norm case. It takes as inputs a matrix $A \in \mathbb{R}^{n \times d}$ with $\mathbf{rank}(A) = \rho$ and a rank parameter $k \ll \rho$, and outputs a set of numbers $\hat{p}_i$ for all $i \in [n]$, namely our approximations to the normalized leverage scores of A with rank parameter k. It is worth noting that $\sum_{i=1}^n \hat{\ell}_i = \left\| Q U_{Q^T A, k} \right\|_F^2 = \left\| U_{Q^T A, k} \right\|_F^2 = k$ and thus the $\hat{p}_i$ sum up to one. The next lemma argues that there exists a matrix $X \in \mathbb{R}^{n \times d}$ of rank k that is sufficiently close to A (in particular, it is a member of $\mathcal{S}_\varepsilon$ with constant probability). Unlike the previous section (the spectral norm case), we will now be able to provide a closed-form formula for this matrix X and, more importantly, the normalized leverage scores of X will be *exactly equal* to the $\hat{p}_i$ returned by our algorithm. Thus, in the parlance of Definition 14, we will get a 1-approximation to the normalized leverage scores of A with rank parameter k.

Lemma 17 (Frobenius Sketch) *Given $A \in \mathbb{R}^{n \times d}$ of rank ρ, a rank parameter k such that $2 \le k < \rho$, and an error parameter ε such that $0 < \varepsilon < 1$, let $\Pi \in \mathbb{R}^{d \times r}$ be a standard Gaussian matrix (with entries selected in i.i.d. trials from $\mathcal{N}(0,1)$) with*

$$r \ge k + \left\lceil \frac{10k}{\varepsilon} + 1 \right\rceil. \tag{18}$$

Let $B = A\Pi$ and let X be as in Equation (19). Then,

$$\mathbf{E}\left[\|A - X\|_F^2\right] \le \left(1 + \frac{\varepsilon}{10}\right)\|A - A_k\|_F^2.$$

The matrix B can be computed in $O\left(ndk\varepsilon^{-1}\right)$ time.

Let

$$X = Q\left(Q^T A\right)_k \in \mathbb{R}^{n \times d}, \tag{19}$$

where $\left(Q^T A\right)_k$ is the best rank-k approximation to the matrix $Q^T A$; from standard linear algebra, $\left(Q^T A\right)_k = U_{Q^T A,k} U_{Q^T A,k}^T Q^T A$. Then, the above lemma is proven in Boutsidis et al. (2011b).[11] Now, since X has rank k, it follows that $\|A - X\|_F^2 \ge \|A - A_k\|_F^2$ and thus we can consider the non-negative random variable $\|A - X\|_F^2 - \|A - A_k\|_F^2$ and apply Markov's inequality to get that

$$\|A - X\|_F^2 - \|A - A_k\|_F^2 \le \varepsilon\|A - A_k\|_F^2$$

holds with probability at least 0.9. Rearranging terms and taking square roots of both sides implies that

$$\|A - X\|_F \le \sqrt{1 + \varepsilon}\|A - A_k\|_F \le (1 + \varepsilon)\|A - A_k\|_F.$$

Thus, $X \in \mathcal{S}_\varepsilon$ with probability at least 0.9. To conclude our proof, recall that Q is an orthonormal basis for the columns of B. From Equation (19),

$$X = Q\left(Q^T A\right)_k = Q U_{Q^T A,k} U_{Q^T A,k}^T Q^T A = Q U_{Q^T A,k} \Sigma_{Q^T A,k} V_{Q^T A,k}^T.$$

In the above, $\Sigma_{Q^T A,k} \in \mathbb{R}^{k \times k}$ is the diagonal matrix containing the top k singular values of $Q^T A$ and $V_{Q^T A,k}^T \in \mathbb{R}^{k \times d}$ is the matrix whose rows are the top k right singular vectors of $Q^T A$. Thus, the left singular vectors of the matrix X are exactly equal to the columns of the orthogonal matrix $Q U_{Q^T A,k}$; it now follows that the $\hat{\ell}_i$ of Equation (17) are the leverage scores of the matrix X and, finally, that the $\hat{p}_i$ returned by the proposed algorithm are the normalized leverage scores of the matrix X.

We briefly discuss the running time of the proposed algorithm. First, we can compute B in $O(ndr)$ time. Then, the computation of Q takes $O(nr^2)$ time. The computation of $Q^T A$ takes $O(ndr)$ time and the computation of $U_{Q^T A}$ takes $O(dr^2)$ time. Thus, the total time is equal to $O\left(ndr + (n + d)r^2\right)$. The following Theorem summarizes the above discussion.

Theorem 18 *Given $A \in \mathbb{R}^{n \times d}$, a rank parameter k, and an accuracy parameter ε, Algorithm 5 computes a set of normalized leverage scores $\hat{p}_i$ that are 1-approximations to the normalized leverage scores of A with rank parameter k with probability at least 0.7. The proposed algorithm runs in $O\left(ndk\varepsilon^{-1} + (n + d)k^2\varepsilon^{-2}\right)$ time.*

6. Discussion

We will conclude with a discussion of our main results in a broader context: understanding the relationship between our main algorithm and a related estimator for the statistical leverage scores; applying our main algorithm to solve under-constrained least squares problems; and implementing variants of the basic algorithm in streaming environments.

---

11. More specifically, the proof may be found in Lemma 33 in Section A.3; note that for our purposes here we set $p = \left\lceil \frac{10k}{\varepsilon} + 1 \right\rceil$.

6.1 A Related Estimator for the Leverage Scores

Magdon-Ismail (2010) presented the following algorithm to estimate the statistical leverage scores: given as input an $n \times d$ matrix A, with $n \gg d$, the algorithm proceeds as follows.

- Compute ΠA, where the $O\left(\frac{n \ln d}{\ln^2 n}\right) \times n$ matrix Π is a SRHT or another FJLT.

- Compute $X = (\Pi A)^\dagger \Pi$.

- For $t = 1, \ldots, n$, compute the estimate $\tilde{w}_t = A_{(t)}^T X^{(t)}$ and set $w_t = \max\left\{\frac{d \ln^2 n}{4n}, \tilde{w}_t\right\}$.

- Return the quantities $\tilde{p}_i = w_i / \sum_{i'=1}^n w_{i'}$, for $i \in [n]$.

Magdon-Ismail (2010) argued that the output $\tilde{p}_i$ achieves an $O(\ln^2 n)$ approximation to all of the (normalized) statistical leverage scores of A in roughly $O(nd^2 / \ln n)$ time. (To our knowledge, prior to our work here, this is the only known estimator that obtains any nontrivial provable approximation to the leverage scores of a matrix in $o(nd^2)$ time.) To see the relationship between this estimator and our main result, recall that

$$\ell_i = e_i^T U U^T e_i = e_i^T A A^\dagger e_i = x_i^T y_i,$$

where the vector $x_i^T = e_i^T A$ is cheap to compute and the vector $y_i = A^\dagger e_i$ is expensive to compute. The above algorithm effectively approximates $y_i = A^\dagger e_i$ via a random projection as $\tilde{y}_i = (\Pi A)^\dagger \Pi e_i$, where Π is a SRHT or another FJLT. Since the estimates $x_i^T \tilde{y}_i$ are not necessarily positive, a truncation at the negative tail, followed by a renormalization step, must be performed in order to arrive at the final estimator returned by the algorithm. This truncation-renormalization step has the effect of inflating the estimates of the small leverage scores by an $O(\ln^2 n)$ factor. By way of comparison, Algorithm 1 essentially computes a sketch of $AA^\dagger$ of the form $A(\Pi A)^\dagger \Pi^T$ that maintains positivity for each of the row norm estimates.

Although both Algorithm 1 and the algorithm of this subsection estimate $AA^\dagger$ by a matrix of the form $A(\Pi A)^\dagger \Pi^T$, there are notable differences. The algorithm of this subsection does not actually compute or approximate AA^T directly; instead, it separates the matrix into two parts and computes the dot product between $e_i^T A$ and $(\Pi A)^\dagger \Pi e_i$. Positivity is sacrificed and this leads to some complications in the estimator; however, the truncation step is interesting, since, despite the fact that the estimates are "biased" (in a manner somewhat akin to what is obtained with "thresholding" or "regularization" procedures), we still obtain provable approximation guarantees. The algorithm of this subsection is simpler (since it uses an application of only one random projection), albeit at the cost of weaker theoretical guarantees and a worse running time than our main algorithm. A direction of considerable practical interest is to evaluate empirically the performance of these two estimators, either for estimating all the leverage scores or (more interestingly) for estimating the largest leverage scores for data matrices for which the leverage scores are quite nonuniform.

6.2 An Application to Under-constrained Least-squares Problems

Consider the following under-constrained least-squares problem:

$$\min_{x \in \mathbb{R}^d} \|Ax - b\|_2, \tag{20}$$

where $A \in \mathbb{R}^{n \times d}$ has much fewer rows than columns, that is, $n \ll d$. It is well-known that we can solve this problem exactly in $O(n^2 d)$ time and that the minimal ℓ_2-norm solution is given by $x_{opt} = A^\dagger b$. For simplicity, let's assume that the input matrix A has full rank (i.e., **rank**$(A) = n$) and thus $\|Ax_{opt} - b\|_2 = 0$.

In this section, we will argue that Algorithm 6 computes a simple, accurate estimator $\tilde{x}_{opt}$ for x_{opt}. In words, Algorithm 6 samples a small number of columns from A (note that the columns of A correspond to variables in our under-constrained problem) and uses the sampled columns to compute $\tilde{x}_{opt}$. However, in order to determine which columns will be included in the sample, the algorithm will make use of the statistical leverage scores of the matrix A^T; more specifically, columns (and thus variables) will be chosen with probability proportional to the corresponding statistical leverage score. We will state Algorithm 6 assuming that these probabilities are parts of the input; the following theorem is our main quality-of-approximation result for Algorithm 6.

Theorem 19 *Let $A \in \mathbb{R}^{n \times d}$ be a full-rank matrix with $n \ll d$; let $\varepsilon \in (0, 0.5]$ be an accuracy parameter; let $\delta \in (0,1)$ be a failure probability; and let $x_{opt} = A^\dagger b$ be the minimal ℓ_2-norm solution to the least-squares problem of Equation (20). Let $p_i \geq 0$, $i \in [d]$, be a set of probabilities satisfying $\sum_{i=1}^{d} p_i = 1$ and*

$$p_i \geq \frac{\beta \|V_{(i)}\|_2^2}{n} \tag{21}$$

for some constant $\beta \in (0,1]$. (Here $V \in \mathbb{R}^{d \times n}$ is the matrix of the right singular vectors of A.) If $\tilde{x}_{opt}$ is computed via Algorithm 6 then, with probability at least $1 - \delta$,

$$\|x_{opt} - \tilde{x}_{opt}\|_2 \leq 2\varepsilon \|x_{opt}\|_2 .$$

Algorithm 6 runs in $O\left(n^3 \varepsilon^{-2} \beta^{-1} \ln\left(n/\varepsilon\beta\delta\right) + nd\right)$ time.

Proof Let the singular value decomposition of the full-rank matrix A be $A = U\Sigma V^T$, with $U \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times n}$, and $V \in \mathbb{R}^{d \times n}$; note that all the diagonal entries of Σ are strictly positive since A has full rank. We can now apply Theorem 4 of Section 6.1 of Drineas et al. (2010b) to get[12] that

$$\left\|I_n - V^T S S^T V\right\|_2 = \left\|V^T V - V^T S S^T V\right\|_2 \leq \varepsilon$$

for our choice of r with probability at least $1 - \delta$. Note that $V^T S \in \mathbb{R}^{n \times r}$ (with $r \geq n$) and let $\sigma_i\left(V^T S\right)$ denote the singular values of $V^T S$ for all $i \in [n]$; the above inequality implies that for all $i \in [n]$

$$\left|1 - \sigma_i^2\left(V^T S\right)\right| \leq \left\|I_n - V^T S S^T V\right\|_2 \leq \varepsilon \leq 0.5.$$

Thus, all the singular values of $V^T S$ are strictly positive and hence $V^T S$ has full rank equal to n. Also, using $\varepsilon \leq 0.5$,

$$\left|1 - \sigma_i^{-2}\left(V^T S\right)\right| \leq \frac{\varepsilon}{1 - \varepsilon} \leq 2\varepsilon. \tag{22}$$

---

12. We apply Theorem 4 of Section 6.1 of Drineas et al. (2010b) with $A = V^T$ and note that $\|V^T\|_F^2 = n \geq 1$, $\|V^T\|_2 = 1$, and $\left(V^T\right)^{(i)} = V_{(i)}$.

We are now ready to prove our theorem:

$$
\begin{aligned}
\left\| x_{opt} - \tilde{x}_{opt} \right\|_2 &= \left\| A^T (AS)^{\dagger T} (AS)^{\dagger} b - A^{\dagger} b \right\|_2 \\
&= \left\| V\Sigma U^T \left(U\Sigma V^T S \right)^{\dagger T} \left(U\Sigma V^T S \right)^{\dagger} b - V\Sigma^{-1} U^T b \right\|_2 \\
&= \left\| \Sigma U^T U\Sigma^{-1} \left(V^T S \right)^{\dagger T} \left(V^T S \right)^{\dagger} \Sigma^{-1} U^T b - \Sigma^{-1} U^T b \right\|_2 \\
&= \left\| \left(V^T S \right)^{\dagger T} \left(V^T S \right)^{\dagger} \Sigma^{-1} U^T b - \Sigma^{-1} U^T b \right\|_2 .
\end{aligned}
$$

In the above derivations we substituted the SVD of A, dropped terms that do not change unitarily invariant norms, and used the fact that $V^T S$ and Σ have full rank in order to simplify the pseudoinverse. Now let $\left(V^T S \right)^{\dagger T} \left(V^T S \right)^{\dagger} = I_n + E$ and note that Equation (22) and the fact that $V^T S$ has full rank imply

$$
\|E\|_2 = \left\| I_n - \left(V^T S \right)^{\dagger T} \left(V^T S \right)^{\dagger} \right\|_2 = \max_{i \in [n]} \left| 1 - \sigma_i^{-2} \left(V^T S \right) \right| \leq 2\varepsilon.
$$

Thus, we conclude our proof by observing that

$$
\begin{aligned}
\left\| x_{opt} - \tilde{x}_{opt} \right\|_2 &= \left\| (I_n + E)\Sigma^{-1} U^T b - \Sigma^{-1} U^T b \right\|_2 \\
&= \left\| E\Sigma^{-1} U^T b \right\|_2 \\
&\leq \|E\|_2 \left\| \Sigma^{-1} U^T b \right\|_2 \\
&\leq 2\varepsilon \left\| x_{opt} \right\|_2 .
\end{aligned}
$$

In the above we used the fact that $\left\| x_{opt} \right\|_2 = \left\| A^{\dagger} b \right\|_2 = \left\| V\Sigma^{-1} U^T b \right\|_2 = \left\| \Sigma^{-1} U^T b \right\|_2$. The running time of the algorithm follows by observing that AS is an $n \times r$ matrix and thus computing its pseudoinverse takes $O(n^2 r)$ time; computing x_{opt} takes an additional $O(nr + dn)$ time. ∎

We conclude the section with a few remarks. First, assuming that ε, β, and δ are constants and $n \ln n = o(d)$, it immediately follows that Algorithm 6 runs in $o(n^2 d)$ time. It should be clear that we can use Theorem 2 and the related Algorithm 1 to approximate the statistical leverage scores, thus bypassing the need to exactly compute them. Second, instead of approximating the statistical leverage scores needed in Algorithm 6, we could use the randomized Hadamard transform (essentially post-multiply A by a randomized Hadamard transform to make all statistical leverage scores uniform). The resulting algorithm could be theoretically analyzed following the lines of Drineas et al. (2010b). It would be interesting to evaluate experimentally the performance of the two approaches in real data.

6.3 Extension to Streaming Environments

In this section, we consider the estimation of the leverage scores and of related statistics when the input data set is so large that an appropriate way to view the data is as a data stream (Muthukrishnan, 2005). In this context, one is interested in computing statistics of the data stream while making one pass (or occasionally a few additional passes) over the data from external storage and using only a small amount of additional space. For an $n \times d$ matrix A, with $n \gg d$, small additional space means that the space complexity only depends *logarithmically* on the high dimension n and *polynomially* on the low dimension d. When we discuss bits of space, we assume that the entries of A can be

---

Input: $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, error parameter $\varepsilon \in (0, .5]$, failure probability δ, and a set of probabilities p_i (for all $i \in [d]$) summing up to one and satisfying Equation (21).

Output: $\tilde{x}_{opt} \in \mathbb{R}^d$.

1. Let $r = \frac{96n}{\beta \varepsilon^2} \ln \left(\frac{96n}{\beta \varepsilon^2 \sqrt{\delta}} \right)$.

2. Let $S \in \mathbb{R}^{d \times r}$ be an all-zeros matrix.

3. **For** $t = 1, \ldots, r$ **do**

   - Pick $i_t \in [d]$ such that $\mathbf{Pr}\,(i_t = i) = p_i$.
   - $S_{i_t, t} = 1/\sqrt{r p_{i_t}}$.

4. Return $\tilde{x}_{opt} = A^T (AS)^{\dagger T} (AS)^{\dagger} b$.

Algorithm 6: Approximately solving under-constrained least squares problems.

discretized to $O(\log n)$ bit integers, though all of our results can be generalized to arbitrary word sizes. The general strategy behind our algorithms is as follows.

- As the data streams by, compute TA, for an appropriate problem-dependent linear sketching matrix T, and also compute ΠA, for a random projection matrix Π.[13]

- After the first pass over the data, compute the matrix R^{-1}, as described in Algorithm 1, corresponding to ΠA (or compute the pseudoinverse of ΠA or the R matrix from any other QR decomposition of A).

- Compute $TAR^{-1}\Pi_2$, for a random projection matrix Π_2, such as the one used by Algorithm 1.

With the procedure outlined above, the matrix T is effectively applied to the rows of $AR^{-1}\Pi_2$, that is, to the sketch of A that has rows with Euclidean norms approximately equal to the row norms of U, and pairwise inner products approximately equal to those in U. Thus statistics related to U can be extracted.

Large Leverage Scores. Given any $n \times d$ matrix A in a streaming setting, it is known how to find the indices of all rows $A_{(i)}$ of A for which $\|A_{(i)}\|_2^2 \geq \tau \|A\|_F^2$, for a parameter τ, and in addition it is known how to compute a $(1 + \varepsilon)$-approximation to $\|A_{(i)}\|_2^2$ for these large rows. The basic idea is to use the notion of ℓ_2-sampling on matrix A, namely, to sample random entries A_{ij} with probability $A_{ij}^2/\|A\|_F^2$. A single entry can be sampled from this distribution in a single pass using $O(\varepsilon^{-2} \log^3(nd))$ bits of space (Monemizadeh and Woodruff, 2010; Andoni et al., 2010). More precisely, these references demonstrate that there is a distribution over $O(d\varepsilon^{-2} \log^3(nd)) \times n$ matrices

---

13. In the offline setting, one would use an SRHT or another FJLT, while in the streaming setting one could use either of the following. If the stream is such that one sees each entire column of A at once, then one could do an FJLT on the column. Alternatively, if one see updates to the individual entries of A in an arbitrary order, then one could apply any sketching matrix, such as those of Achlioptas (2003) or of Dasgupta et al. (2010).

T for which for any fixed matrix $A \in \mathbb{R}^{n \times d}$, there is a procedure which given TA, outputs a sample $(i, j) \in [n] \times [d]$ with probability $(1 \pm \varepsilon) \frac{A_{i,j}^2}{\|A\|_F^2} \pm n^{-O(1)}$. Technically, these references concern sampling from vectors rather than matrices, so $T(A)$ is a linear operator which treats A as a length-nd vector and applies the algorithm of Monemizadeh and Woodruff (2010) and Andoni et al. (2010). However, by simply increasing the number of rows in T by a factor of the small dimension d, we can assume T is left matrix multiplication. By considering the marginal along $[n]$, the probability that $i = a$, for any $a \in [n]$, is

$$(1 \pm \varepsilon) \frac{\|U_{(a)}\|_2^2}{\|U\|_F^2} \pm (nd)^{-O(1)}.$$

By the coupon collector problem, running $O(\tau^{-1} \log \tau^{-1})$ independent copies is enough to find a set containing all rows $A_{(i)}$ for which $\|A_{(i)}\|_2^2 \geq \tau \|A\|_F^2$, and no rows $A_{(i)}$ for which $\|A_{(i)}\|_2^2 < \frac{\tau}{2} \|A\|_F^2$ with probability at least 0.99.

When applied to our setting, we can apply a random projection matrix Π and a linear sketching matrix T which has $O(d\tau^{-1} \varepsilon^{-2} \log^3(n) \log \tau^{-1})$ rows in the following manner. First, TA and ΠA are computed in the first pass over the data; then, at the end of the first pass, we compute R^{-1}; and finally, we compute $TAR^{-1}\Pi_2$, for a random projection matrix Π_2. This procedure effectively applies the matrix T to the rows of $AR^{-1}\Pi_2$, which have norms equal to the row norms of U, up to a factor of $1 + \varepsilon$. The multiplication at the end by Π_2 serves only to speed up the time for processing TAR^{-1}. Thus, by the results of Monemizadeh and Woodruff (2010) and Andoni et al. (2010), we can find all the leverage scores $\|U_{(i)}\|_2^2$ that are of magnitude at least $\tau \|U\|_F^2$ in small space and a single pass over the data. By increasing the space by a factor of $O(\varepsilon^{-2} \log n)$, we can also use the ℓ_2-samples to estimate the norms $\|U_{(i)}\|_2^2$ for the row indices i that we find.

Entropy. Given a distribution ρ, a statistic of ρ of interest is the entropy of this distribution, where the entropy is defined as $H(\rho) = \sum_i \rho(i) \log_2(1/\rho(i))$. This statistic can be approximated in a streaming setting. Indeed, it is known that estimating $H(\rho)$ up to an additive ε can be reduced to $(1 + \tilde{\varepsilon})$-approximation of the ℓ_p-norm of the vector $(\rho(1), \ldots, \rho(n))$, for $O(\log 1/\varepsilon)$ different $p \in (0, 1)$ (Harvey et al., 2008). Here $\tilde{\varepsilon} = \varepsilon/(\log^3 1/\varepsilon \cdot \log n)$. When applied to our setting, the distribution of interest is $\rho(i) = \frac{1}{d}\|U_{(i)}\|_2^2$. To compute the entropy of this distribution, there exist sketching matrices T for providing $(1 + \varepsilon)$-approximations to the quantity $F_p(F_2)$ of an $n \times d$ matrix A, where $F_p(F_2)$ is defined as $\sum_{i=1}^{n} \|A_{(i)}\|_2^{2p}$, using $O(\varepsilon^{-4} \log^2 n \log 1/\varepsilon)$ bits of space (see Theorem 1 of Ganguly et al. (2008)). Thus, to compute the entropy of the leverage score distribution, we can do the following. First, maintain TA and ΠA in the first pass over the data, where T is a sketching matrix for $F_p(F_2)$, $p \in (0, 1)$. At the end of the first pass, compute R^{-1}; and finally, compute $TAR^{-1}\Pi_2$, which effectively applies the $F_p(F_2)$-estimation matrix T to the rows of the matrix $AR^{-1}\Pi_2$. Therefore, by the results of Harvey et al. (2008) and Ganguly et al. (2008), we can compute an estimate ϕ which is within an additive ε of $H(\rho)$ using $O(d\varepsilon^{-4} \log^6 n \log^{14} 1/\varepsilon)$ bits of space and a single pass. We note that it is also possible to estimate $H(\rho)$ up to a multiplicative $1 + \varepsilon$ factor using small, but more, space; see, for example, Harvey et al. (2008).

Sampling Row Identities. Another natural problem is that of obtaining samples of rows of A proportional to their leverage score importance sampling probabilities. To do so, we use ℓ_2-sampling (Monemizadeh and Woodruff, 2010; Andoni et al., 2010) as used above for finding the large leverage scores. First, compute TA and ΠA in the first pass over the data stream; then, compute R^{-1}; and finally, compute TAR^{-1}. Thus, by applying the procedures of Andoni et al. (2010) a total of s times independently, we obtain s samples $i_1, \ldots, i_s$, with replacement, of rows of A proportional

to $\|U_{(i_1)}\|_2^2, \ldots, \|U_{(i_s)}\|_2^2$, that is, to their leverage score. The algorithm requires $O(sd\varepsilon^{-2}\log^4 n)$ bits of space and runs in a single pass. To obtain more than just the row identities $i_1, \ldots, i_s$, for example, to obtain the actual samples, one can read off these rows from A in a second pass over the matrix.

References

D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 557–563, 2006.

N. Ailon and B. Chazelle. The fast Johnson-Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

N. Ailon and E. Liberty. Fast dimension reduction using Rademacher series on dual BCH codes. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–9, 2008.

A. Andoni, R. Krauthgamer, and K. Onak. Streaming algorithms from precision sampling. Technical report, 2010. Preprint: arXiv:1011.1263.

H. Avron, P. Maymounkov, and S. Toledo. Blendenpik: Supercharging LAPACK's least-squares solver. *SIAM Journal on Scientific Computing*, 32:1217–1236, 2010.

C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics*, 57:1214–1229, 2007.

C. Bekas, A. Curioni, and I. Fedulova. Low cost high performance uncertainty quantification. In *Proceedings of the 2nd Workshop on High Performance Computational Finance*, page Article No.: 8, 2009.

P. Bonacich. Power and centrality: A family of measures. *The American Journal of Sociology*, 92 (5):1170–1182, 1987.

C. Boutsidis, M.W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977, 2009.

C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, pages 305–314, 2011a.

C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. Technical report, 2011b. Preprint: arXiv:1103.0995.

E.J. Candes and B. Recht. Exact matrix completion via convex optimization. Technical report, 2008. Preprint: arXiv:0805.4471.

S. Chatterjee and A.S. Hadi. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, 1(3):379–393, 1986.

S. Chatterjee and A.S. Hadi. *Sensitivity Analysis in Linear Regression*. John Wiley & Sons, New York, 1988.

S. Chatterjee, A.S. Hadi, and B. Price. *Regression Analysis by Example*. John Wiley & Sons, New York, 2000.

A. Dasgupta, R. Kumar, and T. Sarlós. A sparse Johnson-Lindenstrauss transform. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pages 341–350, 2010.

P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36:132–157, 2006a.

P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1127–1136, 2006b.

P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:844–881, 2008.

P. Drineas, J. Lewis, and P. Paschou. Inferring geographic coordinates of origin for Europeans using small panels of ancestry informative markers. *PLoS ONE*, 5(8):e11892, 2010a.

P. Drineas, M.W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2010b.

S. Ganguly, M. Bansal, and S. Dube. Estimating hybrid frequency moments of data streams. In *Proceedings of the 2nd Annual International Workshop on Frontiers in Algorithmics*, pages 55–66, 2008.

S. Georgiev and S. Mukherjee, 2011. Unpublished results. (2011).

A. Gittens and M. W. Mahoney, 2012. In preparation. (2012).

G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1996.

N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

N.J.A. Harvey, J. Nelson, and K. Onak. Sketching and streaming entropy via approximation theory. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 489–498, 2008.

D.C. Hoaglin and R.E. Welsch. The hat matrix in regression and ANOVA. *The American Statistician*, 32(1):17–22, 1978.

E. A. Jonckheere, M. Lou, J. Hespanha, and P. Barooah. Effective resistance of Gromov-hyperbolic graphs: Application to asymptotic sensor network problems. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 1453–1458, 2007.

M. Magdon-Ismail. Row sampling for matrix algorithms via a non-commutative Bernstein bound. Technical report, 2010. Preprint: arXiv:1008.0587.

M. W. Mahoney. *Randomized Algorithms for Matrices and Data*. Foundations and trends in machine learning. NOW Publishers, Boston, 2011. Also available at: arXiv:1104.5557.

M.W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proc. Natl. Acad. Sci. USA*, 106:697–702, 2009.

X. Meng, M. A. Saunders, and M. W. Mahoney. LSRN: A parallel iterative solver for strongly over- or under-determined systems. Technical report, 2011. Preprint: arXiv:1109.5981.

M. Monemizadeh and D. P. Woodruff. 1-pass relative-error l_p-sampling with applications. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1143–1160, 2010.

S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Foundations and Trends in Theoretical Computer Science. Now Publishers Inc, Boston, 2005.

M.Z. Nashed, editor. *Generalized Inverses and Applications*. Academic Press, New York, 1976.

M.E.J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27:39–54, 2005.

C.H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: a probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235, 2000.

P. Paschou, E. Ziv, E.G. Burchard, S. Choudhry, W. Rodriguez-Cintron, M.W. Mahoney, and P. Drineas. PCA-correlated SNPs for structure identification in worldwide human populations. *PLoS Genetics*, 3:1672–1686, 2007.

P. Paschou, J. Lewis, A. Javed, and P. Drineas. Ancestry informative markers for fine-scale individual assignment to worldwide populations. *Journal of Medical Genetics*, page doi:10.1136/jmg.2010.078212, 2010.

M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the 8th Annual ACM SIGKDD Conference*, pages 61–70, 2002.

V. Rokhlin and M. Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proc. Natl. Acad. Sci. USA*, 105(36):13212–13217, 2008.

V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2009.

T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 143–152, 2006.

A. Talwalkar and A. Rostamizadeh. Matrix coherence and the Nyström method. In *Proceedings of the 26th Conference in Uncertainty in Artificial Intelligence*, 2010.

P.F. Velleman and R.E. Welsch. Efficient computing of regression diagnostics. *The American Statistician*, 35(4):234–242, 1981.