Fast approximation of matrix coherence and statistical leverage

Petros Drineas DRINEP@CS.RPI.EDU
Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180 USA

Malik Magdon-Ismail MAGDON@CS.RPI.EDU
Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180 USA

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

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

Abstract

The *statistical leverage scores* of a data matrix are the squared row-norms of any matrix whose columns are obtained by orthogonalizing the columns of the data matrix; and, the *coherence* is the largest leverage score. These quantities play an important role in several machine learning algorithms because they capture the key structural nonuniformity of the data matrix that must be dealt with in developing efficient randomized algorithms. Our main result is a randomized algorithm that takes as input an arbitrary $n \times d$ matrix A, with $n \gg d$, and returns, as output, *relative-error* approximations to *all* n of the statistical leverage scores. The proposed algorithm runs 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. This resolves an open question from (Drineas et al., 2006) and (Mohri & Talwalkar, 2011); and our result leads to immediate improvements in coreset-based ℓ_2-regression, the estimation of the coherence of a matrix, and several related low-rank matrix problems. Interestingly, to achieve our result we judiciously apply random projections on *both* sides of A.

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012. Copyright 2012 by the author(s)/owner(s).

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 (Mahoney & 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 & Recht, 2009; Talwalkar & Rostamizadeh, 2010). Statistical leverage scores have a long history in statistical data analysis, where they have been used for outlier detection in regression diagnostics (Hoaglin & Welsch, 1978; Chatterjee & 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 & Drineas, 2009; Sarlós, 2006; Drineas et al., 2010); 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, *e.g.*, the basis provided by the Singular Value Decomposition (SVD) or a basis provided by a QR decomposition, and then use that basis to compute the leverage scores, by taking the norms of the rows.

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 in $O(nd \log n/\epsilon^2)$ time (under assumptions on the precise values of n and d, see Theorem 1 for an exact statement). This is the first algorithm to break the $O(nd^2)$ barrier required by the naïve algorithm, and provide accuracy to within relative error. As a corollary, our algorithm provides a relative-error approximation to the coherence of an arbitrary matrix in the same time. In addition, we discuss several practically-important extensions of the basic idea underlying our main algorithm: computing so-called cross leverage scores; computing leverage scores for "fat" matrices with $n \approx d$ with respect to a low-rank parameter k; and, computing leverage scores in data streaming environments.

1.1. Overview and definitions

We start with the following definition.

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 $\gamma = \max_{i \in \{1,\ldots,n\}} \ell_i$, i.e., it is the largest statistical leverage score of A; and the* (i,j)-cross-leverage scores *c_{ij} are $c_{ij} = \left\langle U_{(i)}, U_{(j)} \right\rangle$, i.e., they are the dot products between the i^{th} row and the j^{th} 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, $\ell_i = \left\|U_{(i)}\right\|_2^2 = \left(UU^T\right)_{ii} = \left(P_A\right)_{ii}$. 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. Similarly, the (i,j)-cross-leverage scores are equal to the off-diagonal elements of this projection matrix, *i.e.*, $c_{ij} = (P_A)_{ij} = \left\langle U_{(i)}, U_{(j)} \right\rangle$. 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.

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 is our main result for Algorithm 1.

Theorem 1. *Let A be a full-rank $n \times d$ matrix, with $n \gg d$; let $\epsilon \in (0, 1/2]$ 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, $\left|\ell_i - \tilde{\ell}_i\right| \le \epsilon \ell_i$ holds for all $i \in \{1, \ldots, n\}$. Assuming $d \le n \le e^d$, the running time of the algorithm is* $O\left(nd \ln\left(d\epsilon^{-1}\right) + nd\epsilon^{-2} \ln n + d^3 \epsilon^{-2} \left(\ln n\right) \left(\ln\left(d\epsilon^{-1}\right)\right)\right)$.

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 the $o(nd^2)$ time.

The following is our main result for Algorithm 2.

Theorem 2. *Let A be a full-rank $n \times d$ matrix, with $n \gg d$; let $\epsilon \in (0, 1/2]$ 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, (1) If $c_{ij}^2 \ge \dfrac{d}{\kappa} + 12\epsilon\ell_i\ell_j$, then (i,j) is returned; if (i,j) is returned, then $c_{ij}^2 \ge \dfrac{d}{\kappa} - 30\epsilon\ell_i\ell_j$. (2) For all pairs (i,j) that are returned, $\tilde{c}_{ij}^2 - 30\epsilon\ell_i\ell_j \le c_{ij}^2 \le \tilde{c}_{ij}^2 + 12\epsilon\ell_i\ell_j$. This algorithm runs in $O(\epsilon^{-2} n \ln n + \epsilon^{-3}\kappa d \ln^2 n)$ time.*

Note that by setting $\kappa = n \ln n$, we can compute all the "large" cross-leverage scores, *i.e.*, 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.

Due to space limitations, numerous details of our analysis are omitted; a full presentation, including full details of the proofs, can be found in the technical report version of this paper (Drineas et al., 2011).

1.3. Significance and related work

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., 2010) and low-rank matrix approximation (Sarlós, 2006; Drineas et al., 2008; Mahoney & 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.

Applications in statistics. The statistical leverage scores equal the diagonal elements of the so-called "hat matrix" (Hoaglin & Welsch, 1978; Chatterjee & Hadi, 1986). As such, they have a natural statistical interpretation in terms of the "leverage" or "influence" associated with each of the data points. Historically, these quantities have been widely-used for outlier identification in diagnostic regression analysis.

Applications in machine learning. Matrix coherence arises in many other applications. For example, (Candes & Recht, 2009) is interested in the problem of matrix completion; (Talwalkar & Rostamizadeh, 2010) is interested in Nyström-based low-rank matrix approximation; (Mohri & Talwalkar, 2011) explicitly ask the question of whether matrix coherence be efficiently and accurately estimated—and thus our main result provides a positive answer to their question. (In other applications, the largest cross-leverage score is called the coherence (Tropp, 2004; Mougeot et al.); thus our results also provide a bound for this quantity.)

2. Linear algebra and fast projections

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 $\rho \le \min\{n, d\}$, in which case the 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 $A^\dagger = V\Sigma^{-1}U^T$.

2.2. The Fast JL Transform (FJLT)

Given $\epsilon > 0$ 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

$$(1-\epsilon)\|x_i\|_2^2 \le \|\Pi x_i\|_2^2 \le (1+\epsilon)\|x_i\|_2^2. \qquad (1)$$

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. Then, the following lemma holds.

Lemma 1 (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 $0 < \epsilon \le 1/2$ be an accuracy parameter. If $r \ge \frac{1}{\epsilon^2}\left(12\ln n + 6\ln\frac{1}{\delta}\right)$ 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 & 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 $\epsilon > 0$, $\Pi \in \mathbb{R}^{r\times n}$ is an ϵ-FJLT for U if: $\left\|I_d - U^T\Pi^T\Pi U\right\|_2 \le \epsilon$; 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., 2006; 2010).

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

$$\mathbf{rank}(\Pi A) = \mathbf{rank}(\Pi U) = \mathbf{rank}(U) = \mathbf{rank}(A), \tag{2}$$

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

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

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. 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. 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 & 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., 2006; 2008; 2010), 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. 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., 2010), restated to fit our notation).

Lemma 3. *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 $r \geq \frac{14^2 d \ln(40nd)}{\epsilon^2} \ln\left(\frac{30^2 d \ln(40nd)}{\epsilon^2}\right)$, then, with probability at least 0.9, Π_{FJLT} is an ϵ-FJLT for U.*

3. Our main algorithmic results

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 Eqn. (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

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

where the first equality follows from the orthogonality of (the columns of) U. The hard part of computing the scores ℓ_i according to Eqn. (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 Eqn. (6).

To get around the bottleneck of $O(nd^2)$ time due to computing $A^\dagger$ in Eqn. (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., 2010) 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 3 with the appropriate choice for r_1). One could approximate the ℓ_i's of Eqn. (6) by

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

where we approximated the $n \times d$ matrix A by the $r_1 \times d$ matrix $\Pi_1 A$. Computing $A\left(\Pi_1 A\right)^\dagger$ in this way takes $O\left(ndr_1\right)$ time, which is not efficient because $r_1 > d$ (from Lemma 3).

To get around this bottleneck, recall that we only need the Euclidean norms of the rows of the matrix

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

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

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

1. Construct $\Pi_1 \in \mathbb{R}^{r_1 \times n}$ to be an $\epsilon$-FJLT for $U$,
   using Lemma 3 with $r_1 = \Omega\left(\frac{d \ln n}{\epsilon^2} \ln\left(\frac{d \ln n}{\epsilon^2}\right)\right)$.
2. Compute $\Pi_1 A \in \mathbb{R}^{r_1 \times d}$ and its SVD, $\Pi_1 A = U_{\Pi_1 A}\Sigma_{\Pi_1 A}V^T_{\Pi_1 A}$.
   Let $R^{-1} = V_{\Pi_1 A}\Sigma^{-1}_{\Pi_1 A} \in \mathbb{R}^{d \times d}$.
   (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 $\epsilon$-JLT for $n^2$ vectors
   (the aforementioned $n$ vectors and their $n^2 - n$ pairwise sums),
   using Lemma 1 with $r_2 = O\left(\epsilon^{-2} \ln n\right)$.
4. Construct the matrix product $\Omega = AR^{-1}\Pi_2$.
5. $\forall i \in [n]$ compute and return $\tilde{\ell}_i = \left\|\Omega_{(i)}\right\|_2^2$.
```

$A\left(\Pi_1 A\right)^\dagger \in \mathbb{R}^{n \times r_1}$. 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 $A\left(\Pi_1 A\right)^\dagger$ (viewed as n vectors in $\mathbb{R}^{r_1}$) and consider the matrix $\Omega = A\left(\Pi_1 A\right)^\dagger \Pi_2$. 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

$$\tilde{\ell}_i = \left\|e_i^T A\left(\Pi_1 A\right)^\dagger \Pi_2\right\|_2^2, \tag{8}$$

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

3.2. Approximating all the leverage scores

Our first main result is Algorithm 1, which takes as input an $n \times d$ matrix A and an error parameter $\epsilon \in (0, 1/2]$, 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 $\hat{A} = U_{\hat{A}}\Sigma_{\hat{A}}V^T_{\hat{A}}$. Let $R^{-1} = V_{\hat{A}}\Sigma^{-1}_{\hat{A}}$ 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. 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.

Lemma 4. *Let R^{-1} be such that $Q = \Pi_1 AR^{-1}$ is an orthogonal matrix with* $\mathbf{rank}(Q) = \mathbf{rank}(\Pi_1 A)$. *Then,* $\left\|(AR^{-1})_{(i)}\right\|_2^2 = \hat{\ell}_i$.

3.3. Approximating large cross-leverage scores

By combining Lemmas 6 and 7 (in Section 4.1 below) with the triangle inequality, one immediately obtains the following lemma.

Lemma 5. *Let Ω be either the sketching matrix constructed by Algorithm 1,* i.e., $\Omega = AR^{-1}\Pi_2$, *or $\Omega = A\left(\Pi_1 A\right)^\dagger \Pi_2$ 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:* $\left|\langle U_{(i)}, U_{(j)}\rangle - \langle \Omega_{(i)}, \Omega_{(j)}\rangle\right| \leq \frac{3\epsilon}{1-\epsilon}\left\|U_{(i)}\right\|_2\left\|U_{(j)}\right\|_2$.

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 $r_2 = O(\epsilon^{-2} \ln n)$. 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 $\epsilon \in (0, 1/2]$, and returns as output a subset of $[n] \times [n]$ and estimates $\tilde{c}_{ij}$ satisfying Theorem 2. The first step of the algorithm is to compute the matrix $\Omega = AR^{-1}\Pi_2$ constructed by Algorithm 1. Then, the algorithm calls Algorithm 3 as a subroutine to compute "heavy hitter" pairs of rows from a matrix.

4. Proofs of our main theorems

4.1. Proof of Theorem 1

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 $\hat{u}_i = e_i^T A(\Pi_1 A)^\dagger$ and $\tilde{u}_i = e_i^T A(\Pi_1 A)^\dagger \Pi_2$. 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.

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

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

Output: The set $\mathcal{H}$ consisting of pairs $(i,j)$ together with estimates $\tilde{c}_{ij}$ satisfying Theorem 2.

1. Compute the $n \times r_2$ matrix $\Omega = AR^{-1}\Pi_2$ from Algorithm 1.
2. Use Algorithm 3 with inputs $\Omega$ and $\kappa' = \kappa(1 + 30d\epsilon)$ to obtain the set $\mathcal{H}$ containing all the $\kappa'$-heavy pairs of $\Omega$.
3. Return the pairs in $\mathcal{H}$ as the $\kappa$-heavy pairs of $A$.
```

Lemma 6. *For $i, j \in [n]$,*

$$\left|\langle U_{(i)}, U_{(j)}\rangle - \langle \hat{u}_i, \hat{u}_j\rangle\right| \le \frac{\epsilon}{1-\epsilon}\left\|U_{(i)}\right\|_2\left\|U_{(j)}\right\|_2. \quad (9)$$

Lemma 7. *For $i, j \in [n]$,*

$$|\langle \hat{u}_i, \hat{u}_j\rangle - \langle \tilde{u}_i, \tilde{u}_j\rangle| \le 2\epsilon\|\hat{u}_i\|_2\|\hat{u}_j\|_2. \quad (10)$$

Lemma 6 states that $\langle \hat{u}_i, \hat{u}_j\rangle$ is an additive error approximation to all the cross-leverage scores ($i \ne j$) and a relative error approximation for the diagonals ($i = j$). Similarly, Lemma 7 shows that these cross-leverage scores are preserved by Π_2. Indeed, with $i = j$, from Lemma 6 we have $|\hat{\ell}_i - \ell_i| \le \frac{\epsilon}{1-\epsilon}\ell_i$, and from Lemma 7 we have $|\hat{\ell}_i - \tilde{\ell}_i| \le 2\epsilon\hat{\ell}_i$. Using the triangle inequality and $\epsilon \le 1/2$:

$$\begin{aligned}\left|\ell_i - \tilde{\ell}_i\right| &= \left|\ell_i - \hat{\ell}_i + \hat{\ell}_i - \tilde{\ell}_i\right| \le \left|\ell_i - \hat{\ell}_i\right| + \left|\hat{\ell}_i - \tilde{\ell}_i\right| \\ &\le \left(\frac{\epsilon}{1-\epsilon} + 2\epsilon\right)\ell_i \le 4\epsilon\ell_i.\end{aligned}$$

The theorem follows after rescaling ϵ.

Running Times. By Lemma 4, we can use $V_{\Pi_1 A}\Sigma_{\Pi_1}^{-1}$ 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 $V_{\Pi_1 A}\Sigma_{\Pi_1}^{-1} \in \mathbb{R}^{d\times d}$. Since $\Pi_2 \in \mathbb{R}^{d\times r_2}$, we obtain $V_{\Pi_1 A}\Sigma_{\Pi_1}^{-1}\Pi_2 \in \mathbb{R}^{d\times r_2}$ 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 $\Omega = AV_{\Pi_1 A}\Sigma_{\Pi_1}^{-1}\Pi_2 \in \mathbb{R}^{n\times r_2}$ takes $O(nr_2)$ time. So, the total running time is the sum of all these operations, which is $O(nd\ln r_1 + ndr_2 + r_1 d^2 + r_2 d^2)$. For our implementations of the ϵ-JLTs and ϵ-FJLTs ($\delta = 0.1$), $r_1 = O\left(\epsilon^{-2} d(\ln n)\left(\ln\left(\epsilon^{-2} d\ln n\right)\right)\right)$ and $r_2 = O(\epsilon^{-2}\ln n)$.

Algorithm 3 Computing heavy pairs of a matrix.

```
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 $\tilde{c}_{ij}^2 = \langle x_i, x_j\rangle^2 \ge \left\|X^T X\right\|_F^2/\kappa$.
1: Compute the norms $\|x_i\|_2$ and sort the rows according to norm, so that $\|x_1\|_2 \le \cdots \le \|x_n\|_2$.
2: $\mathcal{H} \leftarrow \{\}$; $z_1 \leftarrow n$; $z_2 \leftarrow 1$.
3: while $z_2 \le z_1$ 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 $\tilde{c}_{ij}^2 \ge \left\|X^T X\right\|_F^2$ 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 $\mathcal{H}$.
```

It follows that the asymptotic running time is $O\left(nd\ln\left(d\epsilon^{-1}\right) + nd\epsilon^{-2}\ln n + d^3\epsilon^{-2}(\ln n)\left(\ln\left(d\epsilon^{-1}\right)\right)\right)$. To simplify, suppose that $d \le n \le e^d$ and treat ϵ as a constant. Then, the asymptotic running time is $O\left(nd\ln n + d^3(\ln n)(\ln d)\right)$.

4.2. Proof of Theorem 2

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 $\langle x_i, x_j\rangle^2 \ge \frac{1}{\kappa}\left\|X^T X\right\|_F^2$. By the Cauchy-Schwarz inequality, this implies that

$$\|x_i\|_2^2\|x_j\|_2^2 \ge \frac{1}{\kappa}\left\|X^T X\right\|_F^2, \quad (11)$$

so it suffices to find all the pairs (i,j) for which Eqn. (11) holds. We will call such pairs *norm-heavy.* Let s be the number of norm-heavy pairs satisfying Eqn. (11). We first bound the number of such pairs.

Lemma 8. *Using the above notation, $s \le \kappa r$.*

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 $i \leq z_1$ 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 $s \leq \kappa r$ we get the following lemma.

Lemma 9. *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 $\Omega = AV_{\Pi_1 A}\Sigma_{\Pi_1 A}^{-1}\Pi_2 \in \mathbb{R}^{n \times r_2}$, where $r_2 = O(\epsilon^{-2} \ln n)$. 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 5,

$$\langle u_i, u_j \rangle - \Delta \leq \langle \tilde{u}_i, \tilde{u}_j \rangle \leq \langle u_i, u_j \rangle + \Delta, \tag{12}$$

where $\Delta = \frac{3\epsilon}{1-\epsilon}\|u_i\|\|u_j\|$. Given ϵ, κ, assume that for the pair of vectors u_i and u_j

$$\langle u_i, u_j \rangle^2 \geq \frac{1}{\kappa}\left\|U^T U\right\|_F^2 + 12\Delta = \frac{d}{\kappa} + 12\Delta,$$

where $\Delta = \epsilon\|u_i\|^2\|u_j\|^2$, and where the last equality follows from $\left\|U^T U\right\|_F^2 = \|I_d\|_F^2 = d$. By Eqn. (12), after squaring and using $\epsilon < 0.5$,

$$\langle u_i, u_j \rangle^2 - 12\Delta \leq \langle \tilde{u}_i, \tilde{u}_j \rangle^2 \leq \langle u_i, u_j \rangle^2 + 30\Delta, \tag{13}$$

where $\Delta = \epsilon\|u_i\|^2\|u_j\|^2$. Thus, $\langle \tilde{u}_i, \tilde{u}_j \rangle^2 \geq d/\kappa$ and summing Eqn. (13) over all i, j we get $\left\|\Omega^T\Omega\right\|_F^2 \leq d + 30\epsilon d^2$, or, equivalently, $d \geq \frac{\left\|\Omega^T\Omega\right\|_F^2}{1+30d\epsilon}$. We conclude that $\langle u_i, u_j \rangle^2 \geq \frac{d}{\kappa} + 12\epsilon\|u_i\|^2\|u_j\|^2$ implies

$$\langle \tilde{u}_i, \tilde{u}_j \rangle^2 \geq \frac{d}{\kappa} \geq \frac{\left\|\Omega^T\Omega\right\|_F^2}{\kappa(1+30d\epsilon)}. \tag{14}$$

By construction, Algorithm 3 is invoked with $\kappa' = \kappa\left\|\Omega^T\Omega\right\|_F^2/d$ and thus it finds all pairs with $\langle \tilde{u}_i, \tilde{u}_j \rangle^2 \geq \left\|\Omega^T\Omega\right\|_F^2/\kappa' = d/\kappa$. This set contains all pairs for which $\langle u_i, u_j \rangle^2 \geq \frac{d}{\kappa} + 12\epsilon\|u_i\|^2\|u_j\|^2$. Further, since every pair returned satisfies $\langle \tilde{u}_i, \tilde{u}_j \rangle^2 \geq d/\kappa$, by Eqn. (13), $c_{ij} \geq d/\kappa - 30\epsilon\ell_i\ell_j$. This proves the first claim of the Theorem; the second claim follows analogously from Eqn. (13).

Using Lemma 9, the running time of our approach is $O\left(nr_2 + \kappa' r_2^2 + n \ln n\right)$. Since $r_2 = O\left(\epsilon^{-2} \ln n\right)$, and, by Eqn. (14), $\kappa' = \kappa\left\|\Omega^T\Omega\right\|_F^2/d \leq \kappa(1+30d\epsilon)$, the overall running time is $O\left(\epsilon^{-2} n \ln n + \epsilon^{-3}\kappa d \ln^2 n\right)$.

5. Extension to general matrices and streaming environments

Our main result can be extended to the computation of the statistical leverage scores for general "fat" matrices, *i.e.*, matrices $A \in \mathbb{R}^{n \times d}$, where both n and d are large, *e.g.*, $d = n$ or $d = \Theta(n)$, when a rank parameter $k \ll \min\{n, d\}$ 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. As stated, this is an ill-posed problem; and thus the main technical challenge is to deal with this posedness issue. To do so, we note that the leverage scores are used to approximate the matrix in some way; and thus we only care that the estimated leverage scores are a good approximation to the leverage scores of *some* good low-rank approximation to A. Thus, we can define the set of matrices that are good approximations, *e.g.*, with respect to the spectral norm or Frobenius norm, to the best rank k approximation to A, and we can prove that we can efficiently approximate the leverage scores for some matrix in this set.

Our main result can also be extended to estimate the leverage scores and related statistics in data stream environments. In this context, one is interested in computing statistics of the data stream while making one pass 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. The general strategy behind our algorithms is as follows. First, 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. Second, 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). Third, 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 $AR^{-1}\Pi_2$, *i.e.*, 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. For example, in one pass we can: find the indices of all rows of A for with "large" leverage scores and compute a $(1+\epsilon)$-approximation to the leverage scores of these large rows; approximately compute statistics such as the entropy of the distribution of leverage scores of A; and obtain samples of rows of A with proability proportional to their leverage score distribution.

More details can be found in the technical report version of this paper (Drineas et al., 2011).

References

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

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

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

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

Candes, E.J. and Recht, B. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

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

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

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

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

Drineas, P., Magdon-Ismail, M., Mahoney, M. W., and Woodruff, D. P. Fast approximation of matrix coherence and statistical leverage. Technical report, 2011. Preprint: arXiv:1109.3843 (2011).

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

Mahoney, M. W. *Randomized algorithms for matrices and data.* Foundations and Trends in Machine Learning. NOW Publishers, Boston, 2011. Also available at: arXiv:1104.5557.

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

Mohri, M. and Talwalkar, A. Can matrix coherence be efficiently and accurately estimated? In *Proceedings of the 14th International Workshop on Artificial Intelligence and Statistics*, 2011.

Mougeot, M., Picard, D., and Tribouley, K. Learning out of leaders. Technical report. Preprint: arXiv:arXiv:1001.1919 (2010).

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

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

Tropp, J.A. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.