An Improved Approximation Algorithm for the Column Subset Selection Problem [*]

Christos Boutsidis [†] Michael W. Mahoney [‡] Petros Drineas [§]

Abstract

We consider the problem of selecting the "best" subset of *exactly k columns* from an $m \times n$ matrix A. In particular, we present and analyze a novel two-stage algorithm that runs in $O(\min\{mn^2, m^2n\})$ time and returns as output an $m \times k$ matrix C consisting of exactly k columns of A. In the first stage (the *randomized* stage), the algorithm randomly selects $\Theta(k \log k)$ columns according to a judiciously-chosen probability distribution that depends on information in the top-k right singular subspace of A. In the second stage (the *deterministic* stage), the algorithm applies a deterministic column-selection procedure to select and return exactly k columns from the set of columns selected in the first stage. Let C be the $m \times k$ matrix containing those k columns, let P_C denote the projection matrix onto the span of those columns, and let A_k denote the "best" rank-k approximation to the matrix A as computed with the singular value decomposition. Then, we prove that, with probability at least 0.8,

$$\|A - P_C A\|_F \leq \Theta\left(k \log^{1/2} k\right) \|A - A_k\|_F .$$

This Frobenius norm bound is only a factor of $\sqrt{k \log k}$ worse than the best previously existing existential result and is roughly $O(\sqrt{k!})$ better than the best previous algorithmic result (both of Deshpande et al. [11]) for the Frobenius norm version of this Column Subset Selection Problem. We also prove that, with probability at least 0.8,

$$\|A - P_C A\|_2 \leq \Theta\left(k \log^{1/2} k\right) \|A - A_k\|_2 + \Theta\left(k^{3/4} \log^{1/4} k\right) \|A - A_k\|_F .$$

This spectral norm bound is not directly comparable to the best previously existing bounds for the spectral norm version of this Column Subset Selection Problem (such as the ones derived by Gu and Eisenstat in [23]). More specifically, our bound depends on $\|A - A_k\|_F$, whereas previous results depend on $\sqrt{n-k}\,\|A - A_k\|_2$; if these two quantities are comparable, then our bound is asymptotically worse by a $(k \log k)^{1/4}$ factor.

1 Introduction

We consider the problem of selecting the "best" set of *exactly k columns* from an $m \times n$ matrix A. More precisely, we consider the following Column Subset Selection Problem (CSSP):

[*]A conference proceedings version of this paper appeared in [4]. This manuscript presents a modified sampling-based algorithm that fixes a bug in Lemma 4.4 of [4]. This bug also affects the spectral norm bound for the CSSP that was reported in [4]; the Frobenius norm bound remains unaffected.

[†]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, boutsc@cs.rpi.edu.
[‡]Department of Mathematics, Stanford University, Stanford, CA, mmahoney@cs.stanford.edu.
[§]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, drinep@cs.rpi.edu.

Definition 1 ***(The CSSP)*** *Given a matrix $A \in \mathbb{R}^{m \times n}$ and a positive integer k, pick k columns of A forming a matrix $C \in \mathbb{R}^{m \times k}$ such that the residual*

$$\|A - P_C A\|_\xi$$

is minimized over all possible $\binom{n}{k}$ choices for the matrix C. Here, $P_C = CC^+$ denotes the projection onto the k-dimensional space spanned by the columns of C and $\xi = 2$ or F denotes the spectral norm or Frobenius norm.

That is, the goal of the CSSP is to find a subset of exactly k columns of A that "captures" as much of A as possible, with respect to the spectral norm and/or Frobenius norm, in a projection sense. The CSSP has been studied extensively in numerical linear algebra, where it has found applications in, e.g., scientific computing [6]. More recently, a relaxation has been studied in theoretical computer science, where it has been motivated by applications to large scientific and internet data sets [16].

1.1 Complexity of the CSSP

We briefly comment on the complexity of the problem. Clearly, in $O(n^k)$ time we can generate all possible matrices C and thus find the optimal solution in $O(n^k mnk)$ time. However, from a practical perspective, in data analysis applications of the CSSP (see Section 1.2), n is often in the order of hundreds or thousands. Thus, in practice, algorithms whose running time depends exponentially on k are prohibitively slow even if k is, from a theoretical perspective, a constant. Finally, the NP-hardness of the CSSP (assuming k is a function of n) is an open problem. Note, though, that a similar problem, asking for the k columns of the $m \times n$ matrix A that maximize the volume of the parallelepiped spanned by the columns of C, is provably NP-hard [10].

1.2 The CSSP in statistical data analysis

In data applications, where the input matrix A models m objects represented with respect to n features, the CSSP corresponds to unsupervised feature selection. Standard motivations for feature selection include facilitating data visualization, reducing training times, avoiding overfitting, and facilitating data understanding.

Consider, in particular, Principal Components Analysis (PCA), which is the predominant linear dimensionality reduction technique, and which has been widely applied on datasets in all scientific domains, from the social sciences and economics, to biology and chemistry. In words, PCA seeks to map or embed data points from a high dimensional Euclidean space to a low dimensional Euclidean space while keeping all the relevant linear structure intact. PCA is an unsupervised dimensionality reduction technique, with the sole input parameters being the coordinates of the data points and the number of dimensions that will be retained in the embedding (say k), which is typically a constant independent of m and n; often it is $k \ll \{m, n\}$ too. Data analysts often seek a subset of k actual features (that is, k actual columns, as opposed to the k eigenvectors or eigenfeatures returned by PCA) that can accurately reproduce the structure derived by PCA. The CSSP is the obvious optimization problem associated with such unsupervised feature selection tasks.

We should note that similar formulations appeared in [25, 36, 38, 40, 28, 1]. In addition, applications of such ideas include: (*i*) [37], where a "compact CUR matrix decomposition" was applied to static and dynamic data analysis in large sparse graphs; (*ii*) [26, 27, 14], where these ideas were used for compression and classification of hyperspectral medical data and the reconstruction of missing entries from recommendation systems data in order to make high-quality recommendations; and (*iii*) [33], where the concept of "PCA-correlated SNPs" (Single Nucleotide Polymorphisms) was introduced and applied to classify individuals from throughout the world without the need for any prior ancestry information. See [3] for a detailed evaluation of our main algorithm as an unsupervised feature selection strategy in three application domains of modern statistical data analysis (finance, document-term data, and genetics).

1.3 Our main results

We present a novel two-stage algorithm for the CSSP. This algorithm is presented in detail in Section 3 as Algorithm 1. In the first stage of this algorithm (the *randomized stage*), we randomly select $\Theta(k \log k)$ columns of V_k^T, i.e., of the transpose of the $n \times k$ matrix consisting of the top k right singular vectors of A, according to a judiciously-chosen probability distribution that depends on information in the top-k right singular subspace of A. Then, in the second stage (the *deterministic stage*), we apply a deterministic column-selection procedure to select exactly k columns from the set of columns of V_k^T selected by the first stage. The algorithm then returns the corresponding k columns of A. In Section 4 we prove the following theorem.

Theorem 1 *There exists an algorithm (the two-stage Algorithm 1) that approximates the solution to the CSSP. This algorithm takes as input an $m \times n$ matrix A and a positive integer k; it runs in $O(\min\{mn^2, m^2n\})$ time; and it returns as output an $m \times k$ matrix C consisting of exactly k columns of A such that with probability at least 0.8:*

$$\begin{aligned} \|A - P_C A\|_2 &\leq \Theta\left(k \log^{1/2} k\right) \|A - A_k\|_2 + \Theta\left(k^{3/4} \log^{1/4} k\right) \|A - A_k\|_F , \\ \|A - P_C A\|_F &\leq \Theta\left(k \log^{1/2} k\right) \|A - A_k\|_F . \end{aligned}$$

Here, $P_C = CC^+$ denotes a projection onto the column span of the matrix C, and A_k denotes the best rank-k approximation to the matrix A as computed with the singular value decomposition.

Note that we can trivially boost the success probability in the above theorem to $1-\delta$ by repeating the algorithm $O(\log(1/\delta))$ times. Note also that the running time of our algorithm is linear in the larger of the dimensions m and n, quadratic in the smaller one, and independent of k. Thus, it is practically useful and efficient.

To put our results into perspective, we compare them to the best existing results for the CSSP. Prior work provided bounds of the form

$$\|A - P_C A\|_\xi \leq p(k,n) \|A - A_k\|_\xi , \tag{1}$$

where $p(k,n)$ is a polynomial on n and k. For $\xi = 2$, i.e., for the spectral norm, the best previously-known bound for approximating the CSSP is $p(k,n) = \Theta\left(\sqrt{k(n-k)+1}\right)$ [23], while for $\xi = F$, i.e., for the Frobenius norm, the best bound is $p(k,n) = \sqrt{(k+1)!}$ [11]. Both results are algorithmically efficient, running in time polynomial in all three parameters m, n, and k. The former runs in $O(mnk \log n)$ time and the latter runs in $O(mnk + kn)$ time. Our approach provides an algorithmic bound for the Frobenius norm version of the CSSP that is roughly $O(\sqrt{k!})$ better than the best previously-known algorithmic result. It should be noted that [11] also proves that by exhaustively testing all $\binom{n}{k}$ possibilities for the matrix C, the best one will satisfy eqn. (1) with $p(k,n) = \sqrt{k+1}$. Our algorithmic result is only $O(\sqrt{k \log k})$ worse than this existential result. A similar existential result for the spectral norm version of the CSSP is proved in [24] with $p(k,n) = \sqrt{k(n-k)+1}$. Our spectral norm bound depends on $\Theta\left(k^{3/4} \log^{1/4} k\right) \|A - A_k\|_F$. In a worst case setting (e.g., when all the bottom $n-k+1$ singular values of A are equal) this quantity is upper bounded by $\Theta\left((n-k)^{1/2}\,k^{3/4}\log^{1/4}k\right)\,\|A-A_k\|_2$. This is worse than the best results for the spectral norm version of the CSSP by a factor of $\Theta\left(k^{1/4}\log^{1/4}k\right)$.

Finally, we should emphasize that a novel feature of the algorithm that we present in this paper is that it combines in a nontrivial manner recent algorithmic developments in the theoretical computer science community with more traditional techniques from the numerical linear algebra community in order to obtain improved bounds for the CSSP.

2 Background and prior work

2.1 Notation and linear algebra

First, recall that the Θ-notation can be used to denote an asymptotically tight bound: $f(n)\in\Theta(g(n))$ or $f(n)=\Theta(g(n))$ if there exist positive constants c_1, c_2, and n_0 such that $0\le c_1g(n)\le f(n)\le c_2g(n)$ for all $n\ge n_0$. This is similar to the way in which the big-O-notation can be used to denote an asymptotic upper bound: $f(n)=O(g(n))$ if there exist positive constants c and n_0 such that $0\le f(n)\le cg(n)$ for all $n\ge n_0$.

Let $[n]$ denote the set $\{1,2,\ldots,n\}$. For any matrix $A\in\mathbb{R}^{m\times n}$, let $A_{(i)}, i\in[m]$ denote the i-th row of A as a row vector, and let $A^{(j)}, j\in[n]$ denote the j-th column of A as a column vector. In addition, let $\|A\|_F^2=\sum_{i,j}A_{ij}^2$ denote the square of its Frobenius norm, and let $\|A\|_2=\sup_{x\in\mathbb{R}^n,\ x\neq 0}|Ax|_2\,/\,|x|_2$ denote its spectral norm. If $A\in\mathbb{R}^{m\times n}$, then the Singular Value Decomposition (SVD) of A can be written as

$$\begin{aligned} A &= U_A\Sigma_A V_A^T \\ &= \begin{pmatrix} U_k & U_{\rho-k}\end{pmatrix}\begin{pmatrix}\Sigma_k & \mathbf{0}\\ \mathbf{0} & \Sigma_{\rho-k}\end{pmatrix}\begin{pmatrix}V_k^T\\ V_{\rho-k}^T\end{pmatrix}. \end{aligned}$$

In this expression, $\rho\le\min\{m,n\}$ denotes the rank of A, $U_A\in\mathbb{R}^{m\times\rho}$ is an orthonormal matrix, Σ_A is a $\rho\times\rho$ diagonal matrix, and $V_A\in\mathbb{R}^{n\times\rho}$ is an orthonormal matrix. Also, Σ_k denotes the $k\times k$ diagonal matrix containing the top k singular values of A, $\Sigma_{\rho-k}$ denotes the $(\rho-k)\times(\rho-k)$ matrix containing the bottom $\rho-k$ singular values of A, V_k denotes the $n\times k$ matrix whose columns are the top k right singular vectors of A, and $V_{\rho-k}$ denotes the $n\times(\rho-k)$ matrix whose columns are the bottom $\rho-k$ right singular vectors of A, etc.

The $m\times k$ orthogonal matrix U_k consisting of the top k left singular vectors of A is the "best" set of k linear combinations of the columns of A, in the sense that $A_k=P_{U_k}A=U_k\Sigma_kV_k^T$ is the "best" rank k approximation to A. Here, $P_{U_k}=U_kU_k^T$ is a projection onto the k-dimensional space spanned by the columns of U_k. In particular, A_k minimizes $\|A-A'\|_\xi$, for both $\xi=2$ and F, over all $m\times n$ matrices A' whose rank is at most k. We also denote $A_{\rho-k}=U_{\rho-k}\Sigma_{\rho-k}V_{\rho-k}^T$. We will use the notation $\|\cdot\|_\xi$ when writing an expression that holds for both the spectral and the Frobenius norm. We will subscript the norm by 2 and F when writing expressions that hold for one norm or the other. Finally, the Moore-Penrose generalized inverse, or pseudoinverse, of A, denoted by A^+, may be expressed in terms of the SVD as $A^+=V_A\Sigma_A^{-1}U_A^T$.

Finally, we will make frequent use of the following fundamental result from probability theory, known as Markov's inequality [30]. Let X be a random variable assuming non-negative values with expectation $\mathbf{E}[X]$. Then, for all $t>0$, $X\le t\cdot\mathbf{E}[X]$ with probability at least $1-t^{-1}$. We will also need the so-called union bound. Given a set of probabilistic events $\mathcal{E}_1,\mathcal{E}_2,\ldots,\mathcal{E}_n$ holding with respective probabilities $p_1,p_2,\ldots,p_n$, the probability that all events hold (a.k.a., the probability of the union of those events) is upper bounded by $\sum_{i=1}^n p_i$.

2.2 Related prior work

Since solving the CSSP exactly is a hard combinatorial optimization problem, research has historically focused on computing approximate solutions to it. Since $\|A - A_k\|_\xi$ provides an immediate lower bound for $\|A - P_C A\|_\xi$, for $\xi = 2, F$ and for any choice of C, a large number of approximation algorithms have been proposed to select a subset of k columns of A such that the resulting matrix C satisfies

$$\|A - A_k\|_\xi \leq \|A - P_C A\|_\xi \leq p(k,n) \|A - A_k\|_\xi$$

for some function $p(k,n)$. Within the numerical linear algebra community, most of the work on the CSSP has focused on spectral norm bounds and is related to the so-called RANK REVEALING QR (RRQR) FACTORIZATION:

Definition 2 ***(The RRQR factorization)*** *Given a matrix $A \in R^{m\times n}$ $(m \geq n)$ and an integer k $(k \leq n)$, assume partial QR factorizations of the form:*

$$A\Pi = QR = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

where $Q \in R^{m\times n}$ is an orthonormal matrix, $R \in R^{n\times n}$ is upper triangular, $R_{11} \in R^{k\times k}$, $R_{12} \in R^{k\times(n-k)}$, $R_{22} \in R^{(n-k)\times(n-k)}$, and $\Pi \in R^{n\times n}$ is a permutation matrix. The above factorization is called a RRQR factorization if it satisfies

$$\begin{array}{rll} \dfrac{\sigma_k(A)}{p_1(k,n)} \leq & \sigma_{min}(R_{11}) & \leq \sigma_k(A) \\ \sigma_{k+1}(A) \leq & \sigma_{max}(R_{22}) & \leq p_2(k,n)\sigma_{k+1}(A), \end{array}$$

where $p_1(k,n)$ and $p_2(k,n)$ are functions bounded by low degree polynomials in k and n.

The work of Golub on pivoted QR factorizations [21] was followed by much research addressing the problem of constructing an efficient RRQR factorization. Most researchers improved RRQR factorizations by focusing on improving the functions $p_1(k,n)$ and $p_2(k,n)$ in Definition 2. Let Π_k denote the first k columns of a permutation matrix Π. Then, if $C = A\Pi_k$ is an $m \times k$ matrix consisting of k columns of A, it is straightforward to prove that

$$\|A - P_C A\|_\xi = \|R_{22}\|_\xi ,$$

for both $\xi = 2, F$. Thus, in particular, when applied to the spectral norm, it follows that

$$\|A - P_C A\|_2 \leq p_2(k,n)\sigma_{k+1}(A) = p_2(k,n) \|A - A_k\|_2 ,$$

i.e., any algorithm that constructs an RRQR factorization of the matrix A with provable guarantees also provides provable guarantees for the CSSP. See Table 1 for a summary of existing results, and see [19] for a survey and an empirical evaluation of some of these algorithms. More recently, [29, 39] proposed random-projection type algorithms that achieve the same spectral norm bounds as prior work while improving the running time.

Within the theoretical computer science community, much work has followed that of Frieze, Kannan, and Vempala [20] on selecting a small subset of representative columns of A, forming a matrix C, such that the projection of A on the subspace spanned by the columns of C is as close to A as possible. The algorithms from this community are randomized, which means that they come with a failure probability, and focus mainly on the Frobenius norm. It is worth noting that they provide a strong tradeoff between the number of selected columns and the desired approximation

Method	**Reference**		**p(k,n)**	**Time**
Pivoted QR	[Golub, 1965]	[21]	$\sqrt{(n-k)}2^k$	$O(mnk)$
High RRQR	[Foster, 1986]	[18]	$\sqrt{n(n-k)}2^{n-k}$	$O(mn^2)$
High RRQR	[Chan, 1987]	[5]	$\sqrt{n(n-k)}2^{n-k}$	$O(mn^2)$
RRQR	[Hong and Pan, 1992]	[24]	$\sqrt{k(n-k)+k}$	-
Low RRQR	[Chan and Hansen, 1994]	[7]	$\sqrt{(k+1)n}2^{k+1}$	$O(mn^2)$
Hybrid-I RRQR	[Chandr. and Ipsen, 1994]	[8]	$\sqrt{(k+1)(n-k)}$	-
Hybrid-II RRQR		[8]	$\sqrt{(k+1)(n-k)}$	-
Hybrid-III RRQR		[8]	$\sqrt{(k+1)(n-k)}$	-
Algorithm 3	[Gu and Eisenstat, 1996]	[23]	$\sqrt{k(n-k)+1}$	-
Algorithm 4		[23]	$\sqrt{f^2k(n-k)+1}$	$O(kmn\log_f(n))$
DGEQPY	[Bischof and Orti, 1998]	[2]	$O(\sqrt{(k+1)^2(n-k)})$	-
DGEQPX		[2]	$O(\sqrt{(k+1)(n-k)})$	-
SPQR	[Stewart, 1999]	[35]	-	-
PT Algorithm 1	[Pan and Tang, 1999]	[32]	$O(\sqrt{(k+1)(n-k)})$	-
PT Algorithm 2		[32]	$O(\sqrt{(k+1)^2(n-k)})$	-
PT Algorithm 3		[32]	$O(\sqrt{(k+1)^2(n-k)})$	-
Pan Algorithm 2	[Pan, 2000]	[31]	$O(\sqrt{k(n-k)+1})$	-

Table 1: Deterministic RRQR algorithms for the CSSP. A dash implies that either the authors do not provide a running time bound or the algorithm depends exponentially on k. (In addition, $m \geq n$ and $f \geq 1$ for this table.)

accuracy. A typical scenario for these algorithms is that the desired approximation error (see ϵ below) is given as input, and then the algorithm selects the minimum number of appropriate columns in order to achieve this error. One of the most relevant results for this paper is a bound of [11], which states that there exist exactly k columns in any $m \times n$ matrix A such that

$$\left\|A - CC^+A\right\|_F \leq \sqrt{k+1}\,\|A - A_k\|_F\,.$$

Here, C contains exactly k columns of A. The only known algorithm to find these k columns is to try all $\binom{n}{k}$ choices and keep the best. This existential result relies on the so-called volume sampling method [11, 12]. In [12], an adaptive sampling method is used to approximate the volume sampling method and leads to an $O(mnk + kn)$ algorithm which finds k columns of A such that

$$\left\|A - CC^+A\right\|_F \leq \sqrt{(k+1)!}\,\|A - A_k\|_F\,.$$

As mentioned above, much work has also considered algorithms choosing slightly more than k columns. This relaxation provides significant flexibility and improved error bounds. For example, in [12], an adaptive sampling method leads to an $O\left(mn\left(k/\epsilon^2 + k^2\log k\right)\right)$ algorithm, such that

$$\left\|A - CC^+A\right\|_F \leq (1+\epsilon)\,\|A - A_k\|_F$$

holds with high probability for some matrix C consisting of $\Theta\left(k/\epsilon^2 + k^2\log k\right)$ columns of A. Similarly, in [15, 16], Drineas, Mahoney, and Muthukrishnan leverage the subspace sampling method to give an $O(\min\{mn^2, m^2n\})$ algorithm such that

$$\left\|A - CC^+A\right\|_F \leq (1+\epsilon)\,\|A - A_k\|_F \tag{2}$$

holds with high probability if C contains $\Theta(k\log k/\epsilon^2)$ columns of A.

3 A two-stage algorithm for the CSSP

In this section, we present and describe Algorithm 1, our main algorithm for approximating the solution to the CSSP. This algorithm takes as input an $m \times n$ matrix A and a rank parameter k. After an initial setup, the algorithm has two stages: a randomized stage and a deterministic stage. In the *randomized stage*, a randomized procedure is run to select $\Theta(k \log k)$ columns from the $k \times n$ matrix V_k^T, i.e., the transpose of the matrix containing the top-k right singular vectors of A. The columns are chosen by randomly sampling according to a judiciously-chosen nonuniform probability distribution that depends on information in the top-k right singular subspace of A. Then, in the *deterministic stage*, a deterministic procedure is employed to select exactly k columns from the $\Theta(k \log k)$ columns chosen in the randomized stage. The algorithm then outputs exactly k columns of A that correspond to those columns chosen from V_k^T. Theorem 1 states that the projection of A on the subspace spanned by these k columns of A is (up to bounded error) close to the best rank k approximation to A.

3.1 Detailed description of our main algorithm

In more detail, Algorithm 1 first computes a probability distribution $p_1, p_2, \ldots, p_n$ over the set $\{1, \ldots, n\}$, i.e., over the columns of V_k^T, or equivalently over the columns of A. The probability distribution depends on information in the top-k right singular subspace of A. In particular, for all $i \in [n]$, define

$$p_i = \frac{\frac{1}{2}\left\|(V_k)_{(i)}\right\|_2^2}{\sum_{j=1}^n \left\|(V_k)_{(j)}\right\|_2^2} + \frac{\frac{1}{2}\left\|\left(\Sigma_{\rho-k} V_{\rho-k}^T\right)^{(i)}\right\|_2^2}{\sum_{j=1}^n \left\|\left(\Sigma_{\rho-k} V_{\rho-k}^T\right)^{(j)}\right\|_2^2}, \tag{3}$$

and note that $p_i \geq 0$, for all $i \in [n]$, and that $\sum_{i=1}^n p_i = 1$. We will describe the computation of probabilities of this form below.

In the *randomized stage*, Algorithm 1 employs the following randomized column selection algorithm to choose $\Theta(k \log k)$ columns from V_k^T to pass to the second stage. Let c assume the value of eqn. (4). In c independent identically distributed (i.i.d.) trials, the algorithm chooses a column of V_k^T where in each trial the i-th column of V_k^T is kept with probability p_i. Additionally, if the i-th column is kept, then a scaling factor equal to $1/\sqrt{cp_i}$ is kept as well. Thus, at the end of this process, we will be left with c columns of V_k^T and their corresponding scaling factors. Notice that due to random sampling in i.i.d. trials with replacement we might keep a particular column more than once.

In order to conveniently represent the c selected columns and the associated scaling factors, we will use the following sampling matrix formalism. First, define an $n \times c$ sampling matrix S_1 as follows: S_1 is initially empty; at each of the c i.i.d. trials, if the i-th column of V_k^T is selected by the random sampling process, then e_i (an n-vector of all-zeros, except for its i-th entry which is set to one) is appended to S_1. Next, define the $c \times c$ diagonal rescaling matrix D_1 as follows: if the i-th column of V_k^T is selected, then a diagonal entry of D_1 is set to $1/\sqrt{cp_i}$. Thus, we may view the randomized stage as outputting the matrix $V_k^T S_1 D_1$ consisting of a small number of rescaled columns of V_k^T, or simply as outputting S_1 and D_1.

In the *deterministic stage*, Algorithm 1 applies a deterministic column selection algorithm to the output of the first stage in order to choose *exactly k columns* from the input matrix A. To do so, we run the Algorithm 4 of [23] (with the parameter f set to $\sqrt{2}$) on the $k \times c$ matrix $V_k^T S_1 D_1$, i.e., the column-scaled version of the columns of V_k^T chosen in the first stage. Thus, a matrix

$V_k^T S_1 D_1 S_2$ is formed, or equivalently, in the sampling matrix formalism described previously, a new matrix S_2 is constructed. Its dimensions are $c \times k$, since it selects exactly k columns out of the c columns returned after the end of the randomized stage. The algorithm then returns the corresponding k columns of the original matrix A, i.e., after the second stage of the algorithm is complete, the $m \times k$ matrix $C = AS_1S_2$ is returned as the final output.

Input: $m \times n$ matrix A, integer k.
Output: $m \times k$ matrix C with k columns of A.

1. **Initial setup:**
   - Compute the top k right singular vectors of A, denoted by V_k.
   - Compute the sampling probabilities p_i, for $i \in [n]$, using eqn. (3) or eqn. (5).
   - Let
     $$c = 1600c_0^2 k \log\left(800c_0^2 k\right) = \Theta(k \log k). \quad (4)$$
     (Here c_0 is the unspecified constant of Theorem 2.)
2. **Randomized Stage:**
   - For $t = 1, \ldots, c$ (i.i.d. trials) select an integer from $\{1, 2, \ldots, n\}$ where the probability of selecting i is equal to p_i. If i is selected, keep the scaling factor $1/\sqrt{cp_i}$.
   - Form the sampling matrix S_1 and the rescaling matrix D_1 (see text).
3. **Deterministic Stage:**
   - Run Algorithm 4, page 853 of [23] (with the parameter f set to $\sqrt{2}$) on the matrix $V_k^T S_1 D_1$ in order to select exactly k columns of $V_k^T S_1 D_1$, thereby forming the sampling matrix S_2 (see text).
   - Return the corresponding k columns of A, i.e., return $C = AS_1S_2$.

Algorithm 1: A two-stage algorithm for the CSSP.

3.2 Running time analysis

We now discuss the running time of our algorithm. Note that manipulating the probability distribution of eqn. (3) yields:

$$p_i = \frac{\left\|(V_k)_{(i)}\right\|_2^2}{2k} + \frac{\left\|(A)^{(i)}\right\|_2^2 - \left\|\left(AV_kV_k^T\right)^{(i)}\right\|_2^2}{2\left(\|A\|_F^2 - \|AV_kV_k^T\|_F^2\right)}. \quad (5)$$

Thus, knowledge of V_k, i.e., the $n \times k$ matrix consisting of the top-k right singular vectors of A, suffices to compute the p_i's. By eqn. (5), $O(\min\{mn^2, m^2n\})$ time suffices for our theoretical analysis. In practice iterative algorithms could be used to speed up the algorithm. Note also that in order to obtain the Frobenius norm bound of Theorem 1, our theoretical analysis holds if the sampling probabilities are of the form:

$$p_i = \left\| (V_k)_{(i)} \right\|_2^2 / k. \tag{6}$$

That is, the Frobenius norm bound of Theorem 1 holds even if the second term in the sampling probabilities of eqns. (3) and (5) is omitted.

Finally, we briefly comment on a technical constraint of Algorithm 4 of [23]. This algorithm assumes that its input matrix has at least as many rows as columns. However, in our approach, we will apply it on the $k \times c$ matrix $V_k^T S_1 D_1$, which clearly has fewer rows than columns. Thus, prior to applying the aforementioned algorithm, we first pad $V_k^T S_1 D_1$ with $c-k$ all-zero rows, thus making it a square matrix. Let $\Omega = V_k^T S_1 D_1$ and let $\tilde{\Omega}$ be the $c \times c$ matrix after the padding. Eqn. (8) in Theorem 3.2 of [23] (with i set to k and f set to $\sqrt{2}$) implies that $\sigma_k(\tilde{\Omega} S_2) \geq \sigma_k(\tilde{\Omega}) / \left(\sqrt{1+2k(c-k)}\right)$. Clearly, $\sigma_k(\tilde{\Omega} S_2) = \sigma_k(\Omega S_2)$ and $\sigma_k(\tilde{\Omega}) = \sigma_k(\Omega)$. Overall, we get,

$$\sigma_k(V_k^T S_1 D_1 S_2) \geq \frac{\sigma_k\left(V_k^T S_1 D_1\right)}{\sqrt{1+2k(c-k)}},$$

which is the only guarantee that we need in the deterministic step (see Lemma 3). The running time of the deterministic stage of Algorithm 1 is $O(c^2 k \log \sqrt{c})$ time, since the (padded) matrix $V_k^T S_1 D_1$ has c columns and rows.

An important open problem would be to identify other suitable importance sampling probability distributions that avoid the computation of a basis for the top-k right singular subspace.

3.3 Intuition underlying our main algorithm

Intuitively, we achieve improved bounds for the CSSP because we apply the deterministic algorithm to a lower dimensional matrix (the matrix $V_k^T S_1 D_1$ with $\Theta(k \log k)$ columns, as opposed to the matrix A with n columns) in which the columns are "spread out" in a "nice" manner. To see this, note that the probability distribution of eqn. (6), and thus one of the two terms in the probability distribution of eqns. (3) or (5), equals (up to scaling) the diagonal elements of the projection matrix onto the span of the top-k right singular subspace. In diagnostic regression analysis, these quantities have a natural interpretation in terms of *statistical leverage*, and thus they have been used extensively to identify "outlying" data points [9]. Thus, the importance sampling probabilities that we employ in the randomized stage of our main algorithm provide a bias toward more "outlying" columns, which then provide a "nice" starting point for the deterministic stage of our main algorithm. This also provides intuition as to why using importance sampling probabilities of the form of eqn. (6) leads to relative-error low-rank matrix approximations [15, 16].

4 Proof of Theorem 1

In this section, we provide a proof of Theorem 1. We start with an outline of our proof, pointing out conceptual improvements that were necessary in order to obtain improved bounds. An important condition in the first phase of the algorithm is that when we sample columns from the $k \times n$ matrix V_k^T, we obtain a $k \times c$ matrix $V_k^T S_1 D_1$ that does not lose any rank. To do so, we will apply a result from matrix perturbation theory to prove that if $c = \Theta(k \log k)$ (see eqn. (4)) then $\left|\sigma_k^2\left(V_k^T S_1 D_1\right) - 1\right| \leq 1/2$. (See Lemma 1 below.) Then, under the assumption that $V_k^T S_1 D_1$ has full rank, we will prove that the $m \times k$ matrix C returned by the algorithm will satisfy:

$$\|A - P_C A\|_\xi \leq \|A - A_k\|_\xi + \sigma_k^{-1}\left(V_k^T S_1 D_1 S_2\right) \left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\right\|_\xi$$

for both $\xi = 2, F$. (See Lemma 2 below.) Next, we will provide a bound on $\sigma_k^{-1}\left(V_k^T S_1 D_1 S_2\right)$. In order to get a strong accuracy guarantee for the overall algorithm, the deterministic column selection algorithm must satisfy

$$\sigma_k\left(V_k^T S_1 D_1 S_2\right) \geq \frac{\sigma_k\left(V_k^T S_1 D_1\right)}{p(k,c)} > 0,$$

where $p(k,c)$ is a polynomial in both k and c. Thus, for our main theorem, we will employ Algorithm 4 [23] with $f = \sqrt{2}$, which guarantees the above bound with $p(k,c) = \sqrt{2k\,(c-k)+1}$. (See Lemma 3 below.) Finally, we will show, using relatively straightforward matrix perturbation techniques, that $\left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\right\|_\xi$ is not too much more, in a multiplicative sense, than $\|A - A_k\|_\xi$, where we note that the factors differ for $\xi = 2, F$. (See Lemmas 4 and 5 below.) By combining these results, the main theorem will follow.

4.1 The rank of $V_k^T S_1 D_1$

The following lemma provides a bound on the singular values of the matrix $V_k^T S_1 D_1$ computed by the *randomized phase* of Algorithm 1, from which it will follow that the matrix $V_k^T S_1 D_1$ is full rank. To prove the lemma, we employ Theorem 2 of the Appendix (this theorem is a variant of a result of Rudelson and Vershynin in [34]). Note that probabilities of the form of eqn. (6) actually suffice to establish Lemma 1.

Lemma 1 *Let S_1 and D_1 be constructed using Algorithm 1. Then, with probability at least* 0.9,

$$\sigma_k\left(V_k^T S_1 D_1\right) \geq 1/2.$$

In particular, $V_k^T S_1 D_1$ has full rank.

Proof: In order to bound $\sigma_k\left(V_k^T S_1 D_1\right)$, we will bound $\left\|V_k^T S_1 D_1 D_1 S_1^T V_k - I_k\right\|_2$. Towards that end, we will use Theorem 2 with $\beta = 1/2$ and $\epsilon = 1/20$, which results in the value for c in eqn. (4). Note that the sampling probabilities in eqn. (5) satisfy

$$p_i \geq \frac{\left\|(V_k)_{(i)}\right\|_2^2}{2k}.$$

Now Theorem 2 and our construction of S_1 and D_1 guarantee that for c as in eqn. (4)

$$\mathbf{E}\left[\,\left\|V_k^T V_k - V_k^T S_1 D_1 D_1 S_1^T V_k\right\|_2\right] \leq 1/20.$$

We note here that the condition $c_0^2 \|V_k\|_F^2 \geq 4\beta\epsilon^2$ in Theorem 2 is trivially satisfied assuming that c_0 is at least one (given our choices for β, ϵ, and $\|V_k\|_F^2 = k \geq 1$). Using $V_k^T V_k = I_k$ and Markov's inequality we get that with probability at least 0.9,

$$\left\|V_k^T S_1 D_1 D_1 S_1^T V_k - I_k\right\|_2 \;\leq\; 10\,(1/20) = 1/2.$$

Standard matrix perturbation theory results [22] now imply that for all $i = 1, \ldots, k$,

$$\left|\sigma_i^2\left(V_k^T S_1 D_1\right) - 1\right| \leq 1/2.$$

4.2 Bounding the spectral and Frobenius norms of $A - P_C A$

Lemma 2 *Let S_1, D_1, and S_2 be constructed as described in Algorithm 1 and recall that $C = AS_1S_2$. If $V_k^T S_1 D_1$ has full rank, then for $\xi = 2, F$,*

$$\|A - P_C A\|_\xi \leq \|A - A_k\|_\xi + \sigma_k^{-1}\left(V_k^T S_1 D_1 S_2\right)\left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\right\|_\xi .$$

Proof: We seek to bound the spectral and Frobenius norms of $A - P_C A$, where $C = AS_1S_2$ is constructed by Algorithm 1. To do so, first notice that scaling the columns of a matrix (equivalently, post-multiplying the matrix by a diagonal matrix) by any non-zero scale factors does not change the subspace spanned by the columns of the matrix. Thus,

$$\begin{aligned} A - P_C A &= A - (AS_1S_2)(AS_1S_2)^+ A \\ &= A - (AS_1D_1S_2)(AS_1D_1S_2)^+ A \\ &= A - (AS)(AS)^+ A, \end{aligned} \tag{7}$$

where, in the last line, we have introduced the convenient notation $S = S_1 D_1 S_2 \in R^{n\times k}$ that we will use throughout the remainder of this proof. In the sequel we seek to bound the residual

$$\|A - P_C A\|_\xi = \left\|A - (AS)(AS)^+ A\right\|_\xi . \tag{8}$$

First, note that

$$(AS)^+ A = \arg\min_{X\in R^{k\times n}} \|A - ASX\|_\xi .$$

This implies that in eqn. (8) we can replace $(AS)^+A$ with any other $k \times n$ matrix and the equality with an inequality. In particular we replace $(AS)^+A$ with $(A_kS)^+A_k$, where A_k is the best rank-k approximation to A:

$$\begin{aligned} \|A - P_C A\|_\xi &= \left\|A - AS(AS)^+A\right\|_\xi \\ &\leq \left\|A - AS(A_kS)^+A_k\right\|_\xi . \end{aligned}$$

Let $A_{\rho-k} = U_{\rho-k}\Sigma_{\rho-k}V_{\rho-k}^T$. Then, $A = A_k + A_{\rho-k}$ and, using the triangle inequality,

$$\begin{aligned} \|A - P_C A\|_\xi &= \left\|A_k + A_{\rho-k} - (A_k + A_{\rho-k})S(A_kS)^+A_k\right\|_\xi \\ &\leq \underbrace{\left\|A_k - A_kS(A_kS)^+A_k\right\|_\xi}_{\gamma_1} + \underbrace{\left\|A_{\rho-k}\right\|_\xi}_{\gamma_2} + \underbrace{\left\|A_{\rho-k}S(A_kS)^+A_k\right\|_\xi}_{\gamma_3} . \end{aligned} \tag{9}$$

We now bound γ_1, γ_2, and γ_3. First, for γ_1, note that:

$$\begin{aligned} \gamma_1 &= \left\|A_k - A_kS(A_kS)^+A_k\right\|_\xi \\ &= \left\|U_k\Sigma_kV_k^T - U_k\Sigma_k(V_k^TS)(U_k\Sigma_kV_k^TS)^+U_k\Sigma_kV_k^T\right\|_\xi \\ &= \left\|U_k\Sigma_kV_k^T - U_k\Sigma_k(V_k^TS)(V_k^TS)^+(U_k\Sigma_k)^+U_k\Sigma_kV_k^T\right\|_\xi && (10) \\ &= \left\|\Sigma_k - \Sigma_k(V_k^TS)(V_k^TS)^+(U_k\Sigma_k)^+U_k\Sigma_k\right\|_\xi && (11) \\ &= \|\Sigma_k - \Sigma_k\|_\xi = 0. && (12) \end{aligned}$$

In eqn. (10), we replaced $(U_k\Sigma_kV_k^TS)^+$ by $(V_k^TS)^+(U_k\Sigma_k)^+$. This follows since the statement of our lemma assumes that the matrix $V_k^TS_1D_1$ has full rank. Also, the construction of S_2 guarantees that the columns of $V_k^T S_1 D_1$ that are selected in the second stage of Algorithm 1 are linearly independent, and thus the $k \times k$ matrix $V_k^T S = V_k^T S_1 D_1 S_2$ has full rank and is invertible. In eqn. (11), U_k and V_k^T can be dropped without increasing a unitarily invariant norm, while eqn. (12) follows since $V_k^T S$ is a full-rank $k \times k$ matrix. Next, note that $\gamma_2 = \|A_{\rho-k}\|_\xi = \|A - A_k\|_\xi$. Finally, to conclude the proof, we bound γ_3 as follows:

$$\begin{aligned}
\gamma_3 &= \left\|A_{\rho-k} S (A_k S)^+ A_k\right\|_\xi \\
&= \left\|U_{\rho-k}\Sigma_{\rho-k} V_{\rho-k}^T S (U_k \Sigma_k V_k^T S)^+ U_k \Sigma_k V_k^T\right\|_\xi \\
&= \left\|\Sigma_{\rho-k} V_{\rho-k}^T S (V_k^T S)^+\right\|_\xi \qquad (13)\\
&\leq \left\|\Sigma_{\rho-k} V_{\rho-k}^T S\right\|_\xi \left\|\left(V_k^T S\right)^{-1}\right\|_2 \qquad (14)\\
&= \sigma_k^{-1}\left(V_k^T S\right) \left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\right\|_\xi . \qquad (15)
\end{aligned}$$

Eqn. (13) follows by the orthogonality of $U_{\rho-k}$ and V_k and the fact that $V_k^T S$ is a $k \times k$ invertible matrix (see above). Eqn. (14) follows from the fact that for any two matrices X and Y and $\xi = 2, F$, $\|XY\|_\xi \leq \|X\|_\xi \|Y\|_2$. Finally, eqn. (15) follows since $S = S_1 D S_2$ and S_2 is an orthogonal matrix.

4.3 Upper bounds for $\sigma_k^{-1}\left(V_k^T S_1 D_1 S_2\right)$ and $\left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\right\|_\xi$, $\xi = 2, F$

Lemma 3 *Let S_1, D_1, and S_2 be constructed using Algorithm 1. Then, with probability at least* 0.9*,*

$$\sigma_k^{-1}\left(V_k^T S_1 D_1 S_2\right) \leq 2\sqrt{2k\,(c-k)+1}.$$

Proof: From Lemma 1 we know that $\sigma_i\left(V_k^T S_1 D_1\right) \geq 1/2$ holds for all $i = 1, \ldots, k$ with probability at least 0.9. The deterministic construction of S_2 (see Algorithm 4 of [23] with the parameter f set to $\sqrt{2}$) guarantees that

$$\sigma_k(V_k^T S_1 D_1 S_2) \geq \frac{\sigma_k(V_k^T S_1 D_1)}{\sqrt{2k\,(c-k)+1}} \geq \frac{1}{2\sqrt{2k\,(c-k)+1}}.$$

Lemma 4 *($\xi = 2$) If S_1 and D_1 are constructed as described in Algorithm 1, then, with probability at least* 0.9*,*

$$\left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\right\|_2 \leq \|A - A_k\|_2 + \frac{4}{c^{1/4}}\|A - A_k\|_F .$$

Proof: Let $\Gamma = \Sigma_{\rho-k} V_{\rho-k}^T V_{\rho-k} \Sigma_{\rho-k} = \Sigma_{\rho-k}^2$. We manipulate $\left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\right\|_2^2$ as follows:

$$\begin{aligned}
\left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1\right\|_2^2 &= \left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k}\right\|_2 \\
&= ||\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k} - \Gamma + \Gamma||_2 \\
&\leq ||\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k} - \Gamma||_2 + \left\|\Sigma_{\rho-k}^2\right\|_2 \\
&\leq ||\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k} - \Gamma||_F + \left\|\Sigma_{\rho-k}^2\right\|_2 .
\end{aligned}$$

Given our construction of S_1 and D_1 and applying eqn. (9) of Theorem 1 of [13] with $\beta = 1/2$ and $\delta = 0.1$, we get that with probability at least 0.9,

$$\left\|\Sigma_{\rho-k} V_{\rho-k}^T S_1 D_1 D_1 S_1^T V_{\rho-k} \Sigma_{\rho-k} - \Sigma_{\rho-k} V_{\rho-k}^T V_{\rho-k} \Sigma_{\rho-k}\right\|_F \leq \frac{12}{\sqrt{c}}\left\|\Sigma_{\rho-k} V_{\rho-k}^T\right\|_F^2 .$$

Thus, by combining the above results and using $\left\|\Sigma_{\rho-k}^2\right\|_2 = \|A - A_k\|_2^2$ and $\left\|\Sigma_{\rho-k}V_{\rho-k}^T\right\|_F^2 = \|A - A_k\|_F^2$ we get

$$\left\|\Sigma_{\rho-k}V_{\rho-k}^T S_1 D_1\right\|_2^2 \leq \frac{12}{\sqrt{c}} \|A - A_k\|_F^2 + \|A - A_k\|_2^2 .$$

To conclude the proof of the lemma we take the square roots of both sides of the above inequality.

Lemma 5 *($\xi = F$) If S_1 and D_1 are constructed as described in Algorithm 1, then, with probability at least* 0.9*,*

$$\left\|\Sigma_{\rho-k}V_{\rho-k}^T S_1 D_1\right\|_F \leq 4 \|A - A_k\|_F .$$

Proof: It is straightforward to prove that with our construction of S_1 and D_1, the expectation of $\left\|\Sigma_{\rho-k}V_{\rho-k}^T S_1 D_1\right\|_F^2$ is equal to $\left\|\Sigma_{\rho-k}V_{\rho-k}^T\right\|_F^2$. In addition, note that the latter quantity is exactly equal to $\|A - A_k\|_F^2$. Applying Markov's inequality, we get that, with probability at least 0.9,

$$\left\|\Sigma_{\rho-k}V_{\rho-k}^T S_1 D_1\right\|_F^2 \leq 10 \|A - A_k\|_F^2 .$$

Taking square roots of both sides of the above inequality concludes the proof of the lemma.

4.4 Completing the proof of Theorem 1

To prove the Frobenius norm bound of Theorem 1 we combine Lemma 2 (with $\xi = F$) with Lemmas 3 and 5. Thus, we get

$$\begin{aligned}\|A - P_C A\|_F &\leq \|A - A_k\|_F + \left(2\sqrt{2k(c-k)+1}\right)\left(4\|A - A_k\|_F\right) \\ &= \left(1 + 8\sqrt{2k(c-k)+1}\right)\|A - A_k\|_F .\end{aligned}$$

Using $c = \Theta(k \log k)$ immediately derives the Frobenius norm bound of Theorem 1. Notice that Lemma 3 fails with probability at most 0.1 and that Lemma 5 fails with probability at most 0.1; thus, applying the standard union bound, it follows that the Frobenius norm bound of Theorem 1 holds with probability at least 0.8. To prove the spectral norm bound of Theorem 1 we combine Lemma 2 (with $\xi = 2$) with Lemmas 3 and 4. Thus, we get

$$\begin{aligned}\|A - P_C A\|_2 &\leq \|A - A_k\|_2 + \left(2\sqrt{2k(c-k)+1}\right)\left(\|A - A_k\|_2 + \frac{4}{c^{1/4}}\|A - A_k\|_F\right) \\ &= \left(1 + 2\sqrt{2k(c-k)+1}\right)\|A - A_k\|_2 + \frac{8\sqrt{2k(c-k)+1}}{c^{1/4}}\|A - A_k\|_F .\end{aligned}$$

Using $c = \Theta(k \log k)$ immediately derives the spectral norm bound of Theorem 1. Notice that Lemma 3 fails with probability at most 0.1 and that Lemma 4 fails with probability at most 0.1; thus, applying the standard union bound, it follows that the spectral norm bound of Theorem 1 holds with probability at least 0.8.

Acknowledgements

We are grateful to Daniel Spielman and Ilse Ipsen for numerous useful discussions on the results of this paper. We would also like to thank an anonymous reviewer of an earlier version of this manuscript who provided a counterexample to Lemma 4.4 of [4], and thus helped us identify the error in the proof of that lemma.

References

[1] A. Ben-Hur and I. Guyon. Detecting stable clusters using principal component analysis. *Methods in Molecular Biology*, 224:159–182, 2003.

[2] C. H. Bischof and G. Quintana-Ortí. Computing rank-revealing QR factorizations of dense matrices. *ACM Transactions on Mathematical Software*, 24(2):226–253, 1998.

[3] C. Boutsidis, M.W. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *Proceedings of the 14th Annual ACM SIGKDD Conference*, pages 61–69, 2008.

[4] 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.

[5] T.F. Chan. Rank revealing QR factorizations. *Linear Algebra and Its Applications*, 88/89:67–82, 1987.

[6] T.F. Chan and P.C. Hansen. Some applications of the rank revealing QR factorization. *SIAM Journal on Scientific and Statistical Computing*, 13:727–741, 1992.

[7] T.F. Chan and P.C. Hansen. Low-rank revealing QR factorizations. *Numerical Linear Algebra with Applications*, 1:33–44, 1994.

[8] S. Chandrasekaran and I. C. F. Ipsen. On rank-revealing factorizations. *SIAM Journal on Matrix Analysis and Applications*, 15:592–622, 1994.

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

[10] A. Civril and M. Magdon-Ismail. Finding maximum volume sub-matrices of a matrix. Technical Report 07-08, Rennselar Polytechnic Institute Department of Computer Science, Troy, NY, 2007.

[11] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1117–1126, 2006.

[12] A. Deshpande and S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Proceedings of the 10th International Workshop on Randomization and Computation*, pages 292–303, 2006.

[13] 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, 2006.

[14] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 82–90, 2002.

[15] P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *Proceedings of the 10th International Workshop on Randomization and Computation*, pages 316–326, 2006.

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

[17] P. Drineas, M.W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster least squares approximation. Technical report. Preprint: arXiv:0710.1435v3 (2007).

[18] L. V. Foster. Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra and Its Applications*, 74:47–71, 1986.

[19] L. V. Foster and X. Liu. Comparison of rank revealing algorithms applied to matrices with well defined numerical ranks. *Manuscript. 2006.*

[20] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 370–378, 1998.

[21] G. Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7:206–216, 1965.

[22] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1989.

[23] M. Gu and S.C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing*, 17:848–869, 1996.

[24] Y. P. Hong and C. T. Pan. Rank-revealing QR factorizations and the singular value decomposition. *Mathematics of Computation*, 58:213–232, 1992.

[25] W. J. Krzanowski. Selection of variables to preserve multivariate data structure, using principal components. *Applied Statistics*, 36(1):22–33, 1987.

[26] M.W. Mahoney, M. Maggioni, and P. Drineas. Tensor-CUR decompositions for tensor-based data. In *Proceedings of the 12th Annual ACM SIGKDD Conference*, pages 327–336, 2006.

[27] M.W. Mahoney, M. Maggioni, and P. Drineas. Tensor-CUR decompositions for tensor-based data. *SIAM Journal on Matrix Analysis and Applications*, 30:957–987, 2008.

[28] K. Z. Mao. Identifying critical variables of principal components for unsupervised feature selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(2):339–344, 2005.

[29] P.-G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the approximation of matrices. Technical Report YALEU/DCS/TR-1361, Yale University Department of Computer Science, New Haven, CT, June 2006.

[30] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, New York, 1995.

[31] C.-T. Pan. On the existence and computation of rank-revealing LU factorizations. *Linear Algebra and Its Applications*, 316:199–222, 2000.

[32] C. T. Pan and P. T. P. Tang. Bounds on singular values revealed by QR factorizations. *BIT Numerical Mathematics*, 39:740–756, 1999.

[33] 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.

[34] M. Rudelson and R. Vershynin. Sampling from large matrices: an approach through geometric functional analysis. *Journal of the ACM*, 54(4):Article 21, 2007.

[35] G.W. Stewart. Four algorithms for the efficient computation of truncated QR approximations to a sparse matrix. *Numerische Mathematik*, 83:313–323, 1999.

[36] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, 2003.

[37] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. In *Proceedings of the 7th SIAM International Conference on Data Mining*, 2007.

[38] L. Wolf and A. Shashua. Feature selection for unsupervised and supervised inference: The emergence of sparsity in a weight-based approach. *Journal of Machine Learning Research*, 6:1855–1887, 2005.

[39] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. Technical Report YALEU/DCS/TR-1380, Yale University Department of Computer Science, New Haven, CT, 2007.

[40] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1151–1157, 2007.

Appendix

Let $A \in \mathbb{R}^{m \times n}$ be any matrix. Consider the following algorithm (which is essentially the algorithm in page 876 of [16]) that constructs a matrix $C \in \mathbb{R}^{m \times c}$ consisting of c rescaled columns of A. We state Theorem 4 of [17] that provides a bound for the approximation error $\left\|AA^T - CC^T\right\|_2$.

```
Data   : $A \in \mathbb{R}^{m \times n}$, $p_i \geq 0, i \in [n]$ s.t. $\sum_{i \in [n]} p_i = 1$, positive integer $c \leq n$.
Result : $C \in \mathbb{R}^{m \times c}$
Initialize $S \in \mathbb{R}^{m \times c}$ to be an all-zero matrix.
for $t = 1, \ldots, c$ do
    Pick $i_t \in [n]$, where $\mathbf{Prob}\left(i_t = i\right) = p_i$;
    $S_{i_t t} = 1/\sqrt{cp_{i_t}}$;
end
Return $C = AS$;
```

Algorithm 2: The Exactly(c) algorithm.

Theorem 2 *Let $A \in \mathbb{R}^{m \times n}$ with $\|A\|_2 \leq 1$. Construct C using the* Exactly(c) *algorithm and let the sampling probabilities p_i satisfy*

$$p_i \geq \beta \frac{\left\|A^{(i)}\right\|_2^2}{\|A\|_F^2} \tag{16}$$

for all $i \in [n]$ for some constant $\beta \in (0,1]$. Let $\epsilon \in (0,1)$ be an accuracy parameter, assume $c_0^2 \|A\|_F^2 \geq 4\beta\epsilon^2$, and let

$$c = 2\left(\frac{c_0^2 \|A\|_F^2}{\beta\epsilon^2}\right)\log\left(\frac{c_0^2 \|A\|_F^2}{\beta\epsilon^2}\right).$$

(Here c_0 is the unknown constant of Theorem 3.1, p. 8 of [34].) Then,

$$\mathbf{E}\left[\, \left\|AA^T - CC^T\right\|_2\right] \leq \epsilon.$$