Subspace Sampling and Relative-Error Matrix Approximation: Column-Row-Based Methods

Petros Drineas[1], Michael W. Mahoney[2,⋆], and S. Muthukrishnan[3]

[1] Department of Computer Science, RPI
[2] Yahoo Research Labs
[3] Department of Computer Science, Rutgers University

Abstract. Much recent work in the theoretical computer science, linear algebra, and machine learning has considered matrix decompositions of the following form: given an $m \times n$ matrix A, decompose it as a product of three matrices, C, U, and R, where C consists of a small number of columns of A, R consists of a small number of rows of A, and U is a small carefully constructed matrix that guarantees that the product CUR is "close" to A. Applications of such decompositions include the computation of matrix "sketches", speeding up kernel-based statistical learning, preserving sparsity in low-rank matrix representation, and improved interpretability of data analysis methods. Our main result is a randomized, polynomial algorithm which, given as input an $m \times n$ matrix A, returns as output matrices C, U, R such that

$$\|A - CUR\|_F \leq (1+\epsilon)\,\|A - A_k\|_F$$

with probability at least $1-\delta$. Here, A_k is the "best" rank-k approximation (provided by truncating the Singular Value Decomposition of A), and $\|X\|_F$ is the Frobenius norm of the matrix X. The number of columns in C and rows in R is a low-degree polynomial in k, $1/\epsilon$, and $\log(1/\delta)$. Our main result is obtained by an extension of our recent relative error approximation algorithm for ℓ_2 regression from overconstrained problems to general ℓ_2 regression problems. Our algorithm is simple, and it takes time of the order of the time needed to compute the top k right singular vectors of A. In addition, it samples the columns and rows of A via the method of "subspace sampling," so-named since the sampling probabilities depend on the lengths of the rows of the top singular vectors, and since they ensure that we capture entirely a certain subspace of interest.

1 Introduction

1.1 Motivation and Overview

Recent work in the theoretical computer science, linear algebra, and machine learning has considered matrix decompositions of the following form: given an $m \times n$ matrix A, decompose it as a product of three matrices, C, U, and R, where

⋆ Part of this work was done while at the Department of Mathematics, Yale University.

C consists of a few columns of A, R consists of a few rows of A, and U is a small, carefully constructed matrix that guarantees that the product CUR is "close" to A. Applications of such decompositions include constructing "sketches" for large matrices in a pass-efficient manner, matrix reconstruction, speeding up kernel-based statistical learning computations, sparsity-preservation in low-rank approximations, and improved interpretability of data analysis methods. See [6, 7, 21, 12, 11, 19, 20, 1] for examples of applications for matrix decompositions of this form.

Let us consider the application to data analysis methods in more detail. In many applications, the data are represented by a real $m \times n$ matrix A. Such a matrix may arise if the data consist of n objects, each of which is described by m features. The most common compressed representation of A used by data analysts is that obtained by truncating the Singular Value Decomposition at some number $k \ll \min\{m, n\}$ terms, in large part because this provides the "best" rank-k approximation to A when measured with respect to any unitarily invariant matrix norm. However, there is a fundamental difficulty with this representation: the new "dimensions" (the so-called eigencolumns and eigenrows) of A_k are linear combinations of the original dimensions. As such, they are notoriously difficult to interpret in terms of the underlying data and processes generating that data. For example, the vector $[(1/2)$ age - $(1/\sqrt{2})$ height + $(1/2)$ income$]$, being one of the significant uncorrelated "factors" from a dataset of people's features is not particularly informative. From an analyst's point of view, it would be highly preferable to have a low-rank approximation that is nearly as good as that provided by the SVD but that is expressed in terms of a small number of *actual columns* and *actual rows* of a matrix, rather than linear combinations of those columns and rows.

For example, consider recent data analysis work in DNA microarray and DNA Single Nucleotide Polymorphism (SNP) analysis [14, 15, 17]. Researchers interested in analyzing DNA SNP data often model the data as an $m \times n$ matrix A, where m is the number of individuals in the study, n is the number of SNPs being analyzed, and A_{ij} is an encoding of the i-th SNP value for the j-th indivudual. Since biologists do not have an understanding or intuition about the behavior of, e.g., $30,000$ genes or $1,000,000$ SNPs or 1000 individuals, that they do have about a single gene or a single SNP or a single individual, linear algebraic methods have been employed to extract *actual SNPs* from the computed *eigen-SNPs* in order to be used for further analysis [14, 15, 17]. Our problem of approximating a matrix A by CUR is a direct formulation of this problem; in particular, our problem will determine a small number of actual SNPs to serve as a basis to express the remaining SNPs, and a small number of individuals to serve as a basis to express the remaining individuals.

1.2 Review of Linear Algebra

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. The Singular Value Decomposition (SVD) of

A will be denoted by $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times\rho}$, $\Sigma \in \mathbb{R}^{\rho\times\rho}$, $V \in \mathbb{R}^{n\times\rho}$, and where ρ is the rank of A. The "best" rank-k approximation to A (with respect to, e.g., the Frobenius norm, $||A||_F = \sqrt{\sum_{i,j} A_{ij}^2}$) will be denoted by $A_k = U_k\Sigma_k V_k^T$, where $U_k \in \mathbb{R}^{m\times k}$ is the first k columns of U, etc. The SVD and hence the best rank-k approximation of a general matrix A can be computed in $O(\min\{n^2m, nm^2\})$ time, and optimal rank-k approximations to it can be computed more rapidly with, e.g., Lanczos methods. We will use $SVD(A_k)$ to denote the time to compute A_k. For more details on linear algebra, see [2, 10, 13, 16], and for more details on notation and our sampling matrix formalism, see [4, 9].

1.3 Problem Definition

We start with the following definition.

Definition 1. *Let A be an $m\times n$ matrix, let C be an $m\times c$ matrix whose columns consist of a small number c of columns of the matrix A, and let R be an $r\times n$ matrix whose rows consist of a small number r of rows of the matrix A. Then the $m\times n$ matrix A' is a* column-row-based low-rank matrix approximation *to A, or a* CUR matrix approximation*, if it may be explicitly written as $A' = CUR$ for some $c\times r$ matrix U.*

Note that the combined size of C, U and R is $O(mc + rn + cr)$, which is an improvement over A's size of $O(nm)$ when $c, r \ll n, m$.

The quality of a CUR matrix approximation depends on the choice of C and R as well as on the matrix U. We consider the following problem.

Problem 1 ***(Column-row-based low-rank matrix approximation problem).*** *Given a matrix $A \in \mathbb{R}^{m\times n}$, choose a sufficient number of columns and rows of A and construct a matrix U of appropriate dimensions such that*

$$\|A - CUR\|_F \leq (1+\epsilon)\,\|A - A_k\|_F\,. \tag{1}$$

Here, C is a matrix consisting of the chosen columns of A, R is a matrix consisting of the chosen rows of A, and A_k is the best rank k approximation to A. The number of columns of C and rows of R should be a function of k, $1/\epsilon$, and – in the case of randomized algorithms – a failure probability δ. The running time of the algorithms should be a low-degree polynomial in m, n.

Note that is not obvious whether there exist, and if so whether one can efficiently find, a small (depending on k, $1/\epsilon$, and $1/\delta$, but independent of m and n) number of columns and rows that provide such relative-error guarantees.

1.4 "Subspace Sampling" and Our Main Result

Our main result is the following theorem, which asserts the existence of an algorithm to solve Problem 1.

Theorem 1. *There exists a randomized algorithm that solves Problem 1. In the algorithm, $c = O(k^2 \log(1/\delta)/\epsilon^2)$ columns of A are chosen to construct C, and then $r = O(c^2 \log(1/\delta)/\epsilon^2)$ rows of A are chosen to construct R, and the matrix U is a weighted Moore-Penrose inverse of the intersection between C and R. The algorithm satisfies (1) with probability at least $1 - \delta$, it runs in time $O(SVD(A_k))$, and it uses the method of "subspace sampling" to sample columns to form C and rows to form R.*

For the moment, assume that we are given a set of columns, and consider the following theorem.

Theorem 2. *Let $\epsilon \in (0, 1]$. Let an $m \times n$ matrix A and an $m \times c$ matrix C consisting of c columns of A be given. There exists a randomized algorithm that runs in $O(mn)$ time and constructs an $r \times n$ matrix R consisting of $r = O(c^2 \ln(1/\delta)/\epsilon^2)$ rows of A and a $c \times r$ matrix U such that, with probability at least $1 - \delta$*

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

This algorithm is described in Section 2. This result is a CUR matrix approximation that applies to any subset of columns C of the original matrix A, and has relative error with respect to CC^+A, i.e.,the projection of A on the subspace spanned by the columns of C.

Given Theorem 2, in order to establish Theorem 1 it suffices to find a C for which CC^+A is relative-error approximation to the best rank-k approximation provided by the SVD. It is known that such columns *exist* [18, 3], and recently we designed the first *polynomial time algorithm* to find such a C [8]. We summarize these results in the following theorem. See [8] for details.

Theorem 3. *Let $\epsilon \in (0, 1]$. Let an $m \times n$ matrix A and any positive integer k be given, and let A_k be the best rank k approximation to A. There exists a randomized algorithm that runs in $O(SVD(A_k))$ time and selects $c = O(k^2 \ln(1/\delta)/\epsilon^2)$ columns of A such that, if C is the $m \times c$ matrix whose columns are the selected columns of A, then with probability at least $1 - \delta$*

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

Given a matrix A, it follows from Theorem 3 that we can either choose a column matrix C with this relative-error property or a row matrix R that has an analogous relative-error property. But combining those two matrices C and R does not immediately provide a CUR approximation with the relative-error guarantees. However, by combining Theorem 3 with Theorem 2, we establish Theorem 1 and obtain the relative-error CUR approximation.

The bulk of the technical work is the proof of Theorem 2. We will show that, given a matrix A and a set of its columns C, we can choose a set of its rows R such that CW^+R captures almost as much of A as does CC^+A in a relative error sense, where W^+ is a weighted Moore-Penrose generalized inverse of the intersection between C and R. This is obtained by extending our earlier ℓ_2-regression result [9] to a generalized ℓ_2-regression problem defined below. This extension is the main technical contribution of this paper.

The key technical insight that leads to the relative-error guarantees is that the rows are selected by a novel sampling procedure that we call "subspace sampling." Rather than sample rows from the input matrix with a probability distribution that depends on the Euclidean norms of its rows (which gives provable additive-error bounds [4, 5, 6]), in "subspace sampling" we randomly sample rows of the input matrix with a probability distribution that depends on the Euclidean norms of the rows of the top k singular vectors of the input matrix. This allows us to capture entirely a certain subspace of interest. This is required since we will be performing operations such as pseudoinversion that are not well-behaved to missing a dimension, no matter how insignificant its singular vlaue is. This is different than sampling to capture coarse statistics up to an additive error of $\epsilon||A||_F$, and it requires the use of more complex probabilities and more sophisticated analysis. The precise form of sampling is somewhat complicated and is shown in (4) and (11). It is similar to the sampling method we developed recently to solve the l_2 regression problem [9] that we extend here.

1.5 Related Work

To the best of our knowledge, ours is the first CUR matrix approximation with relative error. Previously, the only know CUR matrix approximations had a large additive error $\epsilon \left\|A\right\|_F$ [6, 7]. In fact, previous to our result, it was not even known whether such a relative-error CUR representation existed. Note that in the linear algebra community, there are several algorithms [19, 20, 1, 12, 11] to get C, U, R like ours for low-rank approximation, but none that is comparable in proven guarantees.

2 The Column-Row-Based Low-Rank Approximation

Assume that we are given an $m \times n$ matrix A and any set of c columns of A forming an $m \times c$ matrix C, and consider the following idea for approximating A. The columns of C are a set of "basis vectors" that are, of course, in general neither orthogonal nor normal. Thus, we can express every column of A as a linear combination of the columns of C. If m and n are large and $c = O(1)$, then this is an overconstrained least-squares fit problem. Thus, for all columns $A^{(j)}, j \in [n]$, we can solve

$$\min_{x_j \in \mathbb{R}^c} \left|A^{(j)} - Cx_j\right|_2 \tag{2}$$

in order to find a c-vector of coefficients x_j and get the optimal least-squares fit for $A^{(j)}$. Equivalently, we seek to solve

$$\min_{X \in \mathbb{R}^{c \times n}} \left\|A - CX\right\|_F \tag{3}$$

in order to express A as $A \approx CX_{opt}$, where $X_{opt} = C^+A$ is a $c \times n$ matrix whose columns are the coefficient vectors $x_j, j \in [n]$ that minimize Equation (2).

We will now use a generalization of the ideas in [9] to modify the above approach to get a CUR decomposition for the matrix A, given C. Instead of solving the generalized least squares problem of Equation (3) we will solve a *sampled* version of the problem, constructed as follows:

1. Compute the SVD of C, $C = U_C \Sigma_C V_C^T$, where $U_C \in \mathbb{R}^{m\times\rho}$, $\Sigma_C \in \mathbb{R}^{\rho\times\rho}$, $V_C \in \mathbb{R}^{c\times\rho}$, and ρ is the rank of C.
2. Compute sampling probabilities p_i for all $i \in [m]$:

$$p_i = \frac{(1/3)\left|(U_C)_{(i)}\right|_2^2}{\sum_{j=1}^n \left|(U_C)_{(j)}\right|_2^2} + \frac{(1/3)\left|(U_C)_{(i)}\right|_2 \left|\left(U_C^\perp U_C^{\perp T} A\right)_{(i)}\right|_2}{\sum_{j=1}^n \left|(U_C)_{(j)}\right|_2 \left|\left(U_C^\perp U_C^{\perp T} A\right)_{(j)}\right|_2} + \frac{(1/3)\left|\left(U_C^\perp U_C^{\perp T} A\right)_{(i)}\right|_2^2}{\sum_{j=1}^n \left|\left(U_C^\perp U_C^{\perp T} A\right)_{(j)}\right|_2^2}. \tag{4}$$

   (Notice that $\sum_{i\in[m]} p_i = 1$.)
3. Create an $m \times r$ sampling matrix S and a $r \times r$ diagonal rescaling matrix D, as defined in [9], in r (an input parameter not greater than m) i.i.d. trials, using the p_i of Equation (4).
4. Return as output the $r \times n$ matrix $R = S^T A$ and the $c \times r$ matrix $U = \left(DS^T C\right)^+ D$.

Note that the time required to compute the SVD of C is $O(c^2 m)$, and computing the probabilities p_i of (4) takes an additional $O(cmn)$ time. Overall, the running time of the algorithm is $O(mn)$ since c, r are constants independent of m, n.

In order to obtain some intuition on the construction of U and R, consider the following "sampled and rescaled" version of Equation (3):

$$\min_{X\in\mathbb{R}^{c\times n}} \left\|DS^T A - DS^T CX\right\|_F. \tag{5}$$

Note that in this "sampled and rescaled" problem, the matrix X has the same dimensions as the matrix X of Equation (3), but that the number of constraints in the overconstrained problems has been reduced from m to r. We will see that solving this "sampled and rescaled" problem, and substituting the solution back into the original problem provides a CUR decomposition with a provable error bound. That is, let $\tilde{X}_{opt} = \left(DS^T C\right)^+ DS^T A = UR$ and use $\tilde{X}_{opt}$ as an approximation to X_{opt} (which achieves the optimal value for the full problem of Equation (3)). Then, we will be able to bound the error

$$\left\|A - C\tilde{X}_{opt}\right\|_F = \left\|A - C\underbrace{\left(DS^T C\right)^+ D}_{U}\underbrace{S^T A}_{R}\right\|_F = \|A - CUR\|_F. \tag{6}$$

Here we have let $W = S^T C$ be the $r \times c$ matrix that corresponds to rows of C that are in R, i.e., equivalently, W contains the common elements of C and R. In addition, C consists of a few columns of A, R consists of a few rows of the matrix A, and $U = (DW)^+ D$; note that in general, $(DW)^+ D \neq W^+$.

Rather than proving that this algorithm leads to a choice of U and R such that $\|A - CUR\|_F \leq (1+\epsilon) \|A - CC^+A\|_F$, we establish in the next section a more general result (of independent interest), of which this is a corollary.

3 Approximating Generalized ℓ_2 Regression

3.1 The Generalized ℓ_2 Regression Problem

In this section, we present and analyze a random sampling algorithm to approximate the following generalized ℓ_2 regression (or least-squares fit) problem: given as input a matrix $A \in \mathbb{R}^{m \times n}$ of rank not greater than k (thus, $A = A_k$ in this section) and a target matrix $B \in \mathbb{R}^{m \times p}$, compute

$$\mathcal{Z} = \min_{X \in \mathbb{R}^{n \times p}} \|B - A_k X\|_F . \tag{7}$$

That is, compute the "best" approximation to the matrix B in the basis provided by the matrix $A = A_k$. Also of interest is the computation of matrices that achieve the minimum $\mathcal{Z}$. The "smallest" matrix among those minimizing $\|B - A_k X\|_F$ is

$$X_{opt} = A_k^+ B. \tag{8}$$

Note that we have not placed any constraints on the relationship between m and n. Since we allow $m > n$, $m = n$, and also $m < n$, our approximation algorithm for generalized ℓ_2 regression can be applied to both overconstrained and underconstrained problems. We do, however, constrain the rank of the matrix A (in this section only).

In the special case that $m \gg n$ and $p = 1$, we have the traditional (very overconstrained) ℓ_2 regression (or least-squares fit) problem: given as input a matrix $A \in \mathbb{R}^{m \times d}$ and a target vector $b \in \mathbb{R}^m$, compute $\mathcal{Z} = \min_{x \in \mathbb{R}^d} |b - Ax|_2$. If $m > d$ there are more constraints than variables and the problem is an *overconstrained* least-squares fit problem; in this case, there does not in general exist a vector x such that $Ax = b$. It is well-known that the minimum-length vector among those minimizing $|b - Ax|_2$ is $x_{opt} = A^+ b$, where A^+ denotes the Moore-Penrose generalized inverse of the matrix A.

In [9], we presented a sampling algorithm for this special case. The algorithm of [9] was the first to use SVD-based sampling probabilities similar to those we use in this paper to solve approximately the generalized ℓ_2 regression problem (7) and (8). Generalizing from $m \gg n$ and $p = 1$ to arbitrary m, n, and p constitutes the main technical contribution of this paper. Generalizing the analysis of [9] to $p > 1$ right-hand side vectors is straightforward; on the other hand, generalizing the analysis of [9] from $m \gg n$, i.e., very overconstrained ℓ_2 problems, to general m and n is more subtle.

3.2 Our Main Algorithm and Theorem for This Problem

We present and analyze an algorithm that constructs and solves an induced subproblem of the ℓ_2 regression problem of Equations (7) and (8). Let DS^TA be the $r \times n$ matrix consisting of the sampled and appropriately rescaled rows of the matrix $A = A_k$, and let DS^TB be the matrix consisting of the sampled and appropriately rescaled rows of B. Then consider the problem

$$\tilde{\mathcal{Z}} = \min_{X \in \mathbb{R}^{n \times p}} \left\| DS^TB - DS^TA_kX \right\|_F . \tag{9}$$

The "smallest" matrix $\tilde{X}_{opt} \in \mathbb{R}^{n \times p}$ among those that achieve the minimum value $\tilde{\mathcal{Z}}$ in the *sampled* generalized ℓ_2 regression problem of Equation (9) is

$$\tilde{X}_{opt} = \left(DS^TA\right)^+ DS^TB. \tag{10}$$

Since we will sample a number of rows $r \ll m$ of the original problem, we will compute (10), and thus (9), exactly. Our main theorem, Theorem 4, states that under appropriate assumptions on the original problem and on the sampling probabilities, the computed quantities $\tilde{\mathcal{Z}}$ and $\tilde{X}_{opt}$ will provide very accurate *relative error* approximations to the exact solution $\mathcal{Z}$ and the optimal matrix X_{opt}.

In more detail, our main algorithm for approximating the solution to the generalized ℓ_2 regression problem takes as input an $m \times n$ matrix A, an $m \times p$ matrix B, and a positive integer $r \leq m$. It returns as output a number $\tilde{\mathcal{Z}}$ and a $n \times p$ matrix $\tilde{X}_{opt}$ by doing the following:

1. Compute A_k, the "best" rank-k approximation to A.
2. Compute sampling probabilities p_i for all $i \in [m]$:

$$p_i = \frac{(1/3)\left|(U_{A,k})_{(i)}\right|_2^2}{\sum_{j=1}^n \left|(U_{A,k})_{(j)}\right|_2^2} + \frac{(1/3)\left|(U_{A,k})_{(i)}\right|_2 \left(U_{A,k}^{\perp}{U_{A,k}^{\perp}}^TB\right)_i}{\sum_{j=1}^n \left|(U_{A,k})_{(j)}\right|_2 \left(U_{A,k}^{\perp}{U_{A,k}^{\perp}}^TB\right)_j}$$
$$+ \frac{(1/3)\left(U_{A,k}^{\perp}{U_{A,k}^{\perp}}^TB\right)_i^2}{\sum_{j=1}^n \left(U_{A,k}^{\perp}{U_{A,k}^{\perp}}^TB\right)_j^2}. \tag{11}$$

3. Create an $m \times r$ sampling matrix S and an $r \times r$ diagonal rescaling matrix D, as defined in [9], in r i.i.d. trials, using the p_i of Equation (11).
4. Solve the induced subproblem, i.e., compute and return the number $\tilde{\mathcal{Z}}$ and the $n \times p$ matrix $\tilde{X}_{opt}$ given by (9) and (10), respectively.

The algorithm (implicitly) forms a sampling matrix S, the transpose of which samples with replacement a few rows of A_k and also the corresponding rows of B, and a rescaling matrix D, which is a diagonal matrix scaling the sampled rows of A_k and the elements of B. Since r rows of A_k and the corresponding

r rows of B are sampled, the algorithm randomly samples with replacement r of the m constraints in the original ℓ_2 regression problem. Thus, intuitively, the algorithm approximates the solution of $A_k X \approx B$ with the exact solution of the downsampled problem $DS^T A_k X \approx DS^T B$. Note that it is the space of constraints that is sampled and that the dimension of the unknown matrix X is the same in both problems.

An important aspect of the algorithm will be the nonuniform sampling probabilities (11). Computing these probabilities clearly takes time of the order of computing the best rank-k approximation to the matrix A plus computing the product $U_{A,k}^{\perp} {U_{A,k}^{\perp}}^T B$. Note that the probabilities (4) are a special case of (11).

Theorem 4 below is our main quality-of-approximation result for this generalized ℓ_2 regression problem. This result states that if the matrix achieving the minimum in the sampled problem is substituted back into the original problem then a good approximation to the original generalized ℓ_2 regression problem is obtained.

Theorem 4. *Let $\epsilon \in (0,1]$. Let an $m \times n$ matrix A that has rank no greater than k, an $m \times p$ matrix B, and the sampling probabilities $\{p_i\}_{i=1}^m$ be given. Let $\mathcal{Z}$ and X_{opt} be the solution to the full generalized ℓ_2-regression problem given by (7) and (8), repsectively, and let $\tilde{\mathcal{Z}}$ and $\tilde{X}_{opt}$ be the solution to the sampled generalized ℓ_2 regression problem, given by (9) and (10), respectively. If the sampling probabilities satisfy (11) and if $r \geq O(d^2 \ln(1/\delta)/\epsilon^2)$, then with probability at least $1-\delta$*

$$\left\| B - A_k \tilde{X}_{opt} \right\|_F \leq (1+\epsilon) \left\| B - A_k X_{opt} \right\|_F .$$

By considering the special case where B is any $m \times n$ matrix A, where A is any $m \times c$ matrix consisting of c actual columns of A, and where $k = \text{rank}(C)$, then Theorem 2 follows as a corollary of Theorem 4.

3.3 Proof of Theorem 4

Due to space limitations, the proof is omitted. It is a generalization of the proof of the main theorem of [9]. Alternatively, see [8].

4 Concluding Remarks

We have presented the first known polynomial time algorithm for obtaining a $(1+\epsilon)$ relative error CUR approximation to a given matrix A. It was previously not even known if such a CUR representation exists, and the best known prior work involved large additive error of $\epsilon \|A\|_F$. This problem is of interest in data analysis, and improved bounds will be useful. Further, it is an interesting open problem whether deterministic results can be btained that match our randomized results. Finally, the sampling method we use has found a few applications since we introduced it in [9]; it is of interest to either find other applications or replace it by simpler sampling methods.

References

1. M.W. Berry, S.A. Pulatova, and G.W. Stewart. Computing sparse reduced-rank approximations to sparse matrices. Technical Report UMIACS TR-2004-32 CMSC TR-4589, University of Maryland, College Park, MD, 2004.
2. R. Bhatia. *Matrix Analysis.* Springer-Verlag, New York, 1997.
3. 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.
4. P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *To appear in: SIAM Journal on Computing.*
5. P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *To appear in: SIAM Journal on Computing.*
6. P. Drineas, R. Kannan, and M.W. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. *To appear in: SIAM Journal on Computing.*
7. P. Drineas and M.W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.
8. P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Polynomial time algorithm for column-row based relative-error low-rank matrix approximation. Technical Report 2006-04, DIMACS, March 2006.
9. P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1127–1136, 2006.
10. G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1989.
11. S.A. Goreinov and E.E. Tyrtyshnikov. The maximum-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, 280:47–51, 2001.
12. S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and Its Applications*, 261:1–21, 1997.
13. R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, New York, 1985.
14. F.G. Kuruvilla, P.J. Park, and S.L. Schreiber. Vector algebra in the analysis of genome-wide expression data. *Genome Biology*, 3:research0011.1–0011.11, 2002.
15. Z. Lin and R.B. Altman. Finding haplotype tagging SNPs by use of principal components analysis. *American Journal of Human Genetics*, 75:850–861, 2004.
16. M.Z. Nashed, editor. *Generalized Inverses and Applications.* Academic Press, New York, 1976.
17. P. Paschou, M.W. Mahoney, J.R. Kidd, A.J. Pakstis, S. Gu, K.K. Kidd, and P. Drineas. Intra- and inter-population genotype reconstruction from tagging SNPs. *Manuscript submitted for publication.*
18. L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via iterative sampling. Technical Report MIT-LCS-TR-983, Massachusetts Institute of Technology, Cambridge, MA, March 2005.

19. G.W. Stewart. Four algorithms for the efficient computation of truncated QR approximations to a sparse matrix. *Numerische Mathematik*, 83:313–323, 1999.
20. G.W. Stewart. Error analysis of the quasi-Gram-Schmidt algorithm. Technical Report UMIACS TR-2004-17 CMSC TR-4572, University of Maryland, College Park, MD, 2004.
21. C.K.I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Annual Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, pages 682–688, 2001.