Foundations and Trends$^{®}$ in
Machine Learning
Vol. 3, No. 2 (2010) 123–224

DOI: 10.1561/2200000035

Randomized Algorithms for Matrices and Data

By Michael W. Mahoney

Contents

Foundations and Trends® in
Machine Learning
Vol. 3, No. 2 (2010) 123–224

DOI: 10.1561/2200000035

Randomized Algorithms for Matrices and Data

Michael W. Mahoney

Department of Mathematics, Stanford University, Stanford, CA 94305, USA, mmahoney@cs.stanford.edu

Abstract

Randomized algorithms for very large matrix problems have received a great deal of attention in recent years. Much of this work was motivated by problems in large-scale data analysis, largely since matrices are popular structures with which to model data drawn from a wide range of application domains, and this work was performed by individuals from many different research communities. While the most obvious benefit of randomization is that it can lead to faster algorithms, either in worst-case asymptotic theory and/or numerical implementation, there are numerous other benefits that are at least as important. For example, the use of randomization can lead to simpler algorithms that are easier to analyze or reason about when applied in counterintuitive settings; it can lead to algorithms with more interpretable output, which is of interest in applications where analyst time rather than just computational time is of interest; it can lead implicitly to regularization and more robust output; and randomized algorithms can often be organized to exploit modern computational architectures better than classical numerical methods.

This monograph will provide a detailed overview of recent work on the theory of randomized matrix algorithms as well as the application of those ideas to the solution of practical problems in large-scale data analysis. Throughout this review, an emphasis will be placed on a few simple core ideas that underlie not only recent theoretical advances but also the usefulness of these tools in large-scale data applications. Crucial in this context is the connection with the concept of statistical leverage. This concept has long been used in statistical regression diagnostics to identify outliers; and it has recently proved crucial in the development of improved worst-case matrix algorithms that are also amenable to high-quality numerical implementation and that are useful to domain scientists. This connection arises naturally when one explicitly decouples the effect of randomization in these matrix algorithms from the underlying linear algebraic structure. This decoupling also permits much finer control in the application of randomization, as well as the easier exploitation of domain knowledge.

Most of the review will focus on random sampling algorithms and random projection algorithms for versions of the linear least-squares problem and the low-rank matrix approximation problem. These two problems are fundamental in theory and ubiquitous in practice. Randomized methods solve these problems by constructing and operating on a randomized sketch of the input matrix A — for random sampling methods, the sketch consists of a small number of carefully-sampled and rescaled columns/rows of A, while for random projection methods, the sketch consists of a small number of linear combinations of the columns/rows of A. Depending on the specifics of the situation, when compared with the best previously-existing deterministic algorithms, the resulting randomized algorithms have worst-case running time that is asymptotically faster; their numerical implementations are faster in terms of clock-time; or they can be implemented in parallel computing environments where existing numerical algorithms fail to run at all. Numerous examples illustrating these observations will be described in detail.

1

Introduction

This monograph will provide a detailed overview of recent work on the theory of *randomized matrix algorithms* as well as the application of those ideas to the solution of practical problems in large-scale data analysis. By "randomized matrix algorithms," we refer to a class of recently-developed random sampling and random projection algorithms for ubiquitous linear algebra problems such as least-squares regression and low-rank matrix approximation. These and related problems are ubiquitous since matrices are fundamental mathematical structures for representing data drawn from a wide range of application domains. Moreover, the widespread interest in randomized algorithms for these problems arose due to the need for principled algorithms to deal with the increasing size and complexity of data that are being generated in many of these application areas.

Not surprisingly, algorithmic procedures for working with matrix-based data have been developed from a range of diverse perspectives by researchers from a wide range of areas — including, e.g., researchers from theoretical computer science (TCS), numerical linear algebra (NLA), statistics, applied mathematics, data analysis, and machine learning, as well as domain scientists in physical and biological sciences — and in many of these cases they have drawn strength from their domain-specific insight. Although this has been great for the development of the area, and for the "technology transfer" of theoretical ideas to practical applications, the technical aspects of dealing with any one of those areas has obscured for many the simplicity and generality of some of the underlying ideas; thus leading researchers to fail to appreciate the underlying connections and the significance of contributions by researchers outside their own area. Thus, rather than focusing on the technical details of proving worst-case bounds or of providing high-quality numerical implementations or of relating to traditional machine learning tools or of using these algorithms in a particular physical or biological domain, in this review we will focus on highlighting for a broad audience the simplicity and generality of some core ideas — ideas that are often obscured but that are fruitful for using these randomized algorithms in large-scale data applications. To do so, we will focus on two fundamental and ubiquitous matrix problems — least-squares approximation and low-rank matrix approximation — that have been at the center of these recent developments.

The work we will review here had its origins within TCS. In this area, one typically considers a particular well-defined problem, and the goal is to prove bounds on the running time and quality-of-approximation guarantees for algorithms for that particular problem that hold for "worst-case" input. That is, the bounds should hold for *any* input matrix, independent of any "niceness" assumptions such as, e.g., that the elements of the matrix satisfy some smoothness or normalization condition or that the spectrum of the matrix satisfies some decay condition. Clearly, the generality of this approach means that the bounds will be suboptimal — and thus can be improved — in any particular application where stronger assumptions can be made about the input. Importantly, though, it also means that the underlying algorithms and techniques will be broadly applicable even in situations where such assumptions do not apply.

An important feature in the use of randomized algorithms in TCS more generally is that one must identify and then algorithmically deal with relevant "non-uniformity structure" in the data.[1] For the randomized matrix algorithms to be reviewed here and that have proven useful recently in NLA and large-scale data analysis applications, the relevant non-uniformity structure is defined by the so-called *statistical leverage scores*. Defined more precisely below, these leverage scores are basically the diagonal elements of the projection matrix onto the dominant part of the spectrum of the input matrix. As such, they have a long history in statistical data analysis, where they have been used for outlier detection in regression diagnostics. More generally, and very importantly for practical large-scale data applications of recently-developed randomized matrix algorithms, these scores often have a very natural interpretation in terms of the data and processes generating the data. For example, they can be interpreted in terms of the leverage or influence that a given data point has on, say, the best low-rank matrix approximation; and this often has an interpretation in terms of high-degree nodes in data graphs, very small clusters in noisy data, coherence of information, articulation points between clusters, etc.

Historically, although the first generation of randomized matrix algorithms (to be described in Section 3) achieved what is known as additive-error bounds and were extremely fast, requiring just a few passes over the data from external storage, these algorithms did *not* gain a foothold in NLA and only heuristic variants of them were used in machine learning and data analysis applications. In order to "bridge the gap" between NLA, TCS, and data applications, much finer control over the random sampling process was needed. Thus, in the second generation of randomized matrix algorithms (to be described in Sections 4 and 5) that *has* led to high-quality numerical implementations

[1] For example, for those readers familiar with Markov chain-based Monte Carlo algorithms as used in statistical physics, this non-uniformity structure is given by the Boltzmann distribution, in which case the algorithmic question is how to sample efficiently with respect to it as an importance sampling distribution without computing the intractable partition function. Of course, if the data are sufficiently nice (or if they have been sufficiently preprocessed, or if sufficiently strong assumptions are made about them, etc.), then that non-uniformity structure might be uniform, in which case simple methods like uniform sampling might be appropriate — but this is far from true in general, either in worst-case theory or in practical applications.

and useful machine learning and data analysis applications, two key developments were crucial.

- **Decoupling the randomization from the linear algebra.** This was originally implicit within the analysis of the second generation of randomized matrix algorithms, and then it was made explicit. By making this decoupling explicit, not only were improved quality-of-approximation bounds achieved, but also *much* finer control was achieved in the application of randomization. For example, it permitted easier exploitation of domain expertise, in both numerical analysis and data analysis applications.
- **Importance of statistical leverage scores.** Although these scores have been used historically for outlier detection in statistical regression diagnostics, they have also been crucial in the recent development of randomized matrix algorithms. Roughly, the best random sampling algorithms use these scores to construct an importance sampling distribution to sample with respect to; and the best random projection algorithms rotate to a basis where these scores are approximately uniform and thus in which uniform sampling is appropriate.

As will become clear, these two developments are very related. For example, once the randomization was decoupled from the linear algebra, it became nearly obvious that the “right” importance sampling probabilities to use in random sampling algorithms are those given by the statistical leverage scores, and it became clear how to improve the analysis and numerical implementation of random projection algorithms. It is remarkable, though, that statistical leverage scores define the non-uniformity structure that is relevant not only to obtain the strongest worst-case bounds, but also to lead to high-quality numerical implementations (by numerical analysts) as well as algorithms that are useful in downstream scientific applications (by machine learners and data analysts).

Most of this review will focus on random sampling algorithms and random projection algorithms for versions of the linear least-squares problem and the low-rank matrix approximation problem. Here is a brief summary of some of the highlights of what follows.

- **Least-squares approximation.** Given an $m \times n$ matrix A, with $m \gg n$, and an m-dimensional vector b, the overconstrained least-squares approximation problem looks for the vector $x_{opt} = \mathrm{argmin}_x ||Ax - b||_2$. This problem typically arises in statistical models where the rows of A and elements of b correspond to constraints and the columns of A and elements of x correspond to variables. Classical methods, including the Cholesky decomposition, versions of the QR decomposition, and the Singular Value Decomposition, compute a solution in $O(mn^2)$ time. Randomized methods solve this problem by constructing a randomized sketch of the matrix A — for random sampling methods, the sketch consists of a small number of carefully-sampled and rescaled rows of A (and the corresponding elements of b), while for random projection methods, the sketch consists of a small number of linear combinations of the rows of A and elements of b. If one then solves the (still overconstrained) subproblem induced on the sketch, then very fine relative-error approximations to the solution of the original problem are obtained. In addition, for a wide range of values of m and n, the running time is $o(mn^2)$ — for random sampling algorithms, the computational bottleneck is computing appropriate importance sampling probabilities, while for random projection algorithms, the computational bottleneck is implementing the random projection operation. Alternatively, if one uses the sketch to compute a preconditioner for the original problem, then very high-precision approximations can be obtained by then calling classical numerical iterative algorithms. Depending on the specifics of the situation, these numerical implementations run in $o(mn^2)$ time; they are faster in terms of clock-time than the best previously-existing deterministic numerical implementations; or they can be implemented in parallel computing environments where existing numerical algorithms fail to run at all.

- **Low-rank matrix approximation.** Given an $m \times n$ matrix A and a rank parameter k, the low-rank matrix approximation problem is to find a good approximation to A of rank $k \ll \min\{m,n\}$. The Singular Value Decomposition provides the best rank-k approximation to A, in the sense that by projecting A onto its top k left or right singular vectors, then one obtains the best approximation to A with respect to the spectral and Frobenius norms. The running time for classical low-rank matrix approximation algorithms depends strongly on the specifics of the situation — for dense matrices, the running time is typically $O(mnk)$; while for sparse matrices, classical Krylov subspace methods are used. As with the least-squares problem, randomized methods for the low-rank matrix approximation problem construct a randomized sketch — consisting of a small number of either actual columns or linear combinations of columns — of the input A, and then this sketch is manipulated depending on the specifics of the situation. For example, random sampling methods can use the sketch directly to construct relative-error low-rank approximations such as CUR decompositions that approximate A based on a small number of actual columns of the input matrix. Alternatively, random projection methods can improve the running time for dense problems to $O(mn\log k)$; and while they only match the running time for classical methods on sparse matrices, they lead to more robust algorithms that can be reorganized to exploit parallel computing architectures.

These two problems are the main focus of this review since they are both fundamental in theory and ubiquitous in practice and since in both cases novel theoretical ideas have already yielded practical results. Although not the main focus of this review, other related matrix-based problems to which randomized methods have been applied will be referenced at appropriate points.

Clearly, when a very new paradigm is compared with very well-established methods, a naïve implementation of the new ideas will perform poorly by traditional metrics. Thus, in both data analysis and numerical analysis applications of this randomized matrix algorithm paradigm, the best results have been achieved when coupling closely with more traditional methods. For example, in data analysis applications, this has meant working closely with geneticists and other domain experts to understand how the non-uniformity structure in the data is useful for their downstream applications. Similarly, in scientific computation applications, this has meant coupling with traditional numerical methods for improving quantities like condition numbers and convergence rates. When coupling in this manner, however, qualitatively improved results have *already* been achieved. For example, in their empirical evaluation of the random projection algorithm for the least-squares approximation problem, to be described in Sections 4.4 and 4.5 below, Avron, Maymounkov, and Toledo [9] began by observing that "Randomization is arguably the most exciting and innovative idea to have hit linear algebra in a long time;" and since their implementation "beats LAPACK's[2] direct dense least-squares solver by a large margin on essentially any dense tall matrix," they concluded that their empirical results "show the potential of random sampling algorithms and suggest that random projection algorithms should be incorporated into future versions of LAPACK."

The remainder of this review will cover these topics in greater detail. To do so, we will start in Section 2 with a few motivating applications from one scientific domain where these randomized matrix algorithms have already found application, and we will describe in Section 3 general background on randomized matrix algorithms, including precursors to those that are the main subject of this review. Then, in the next two sections, we will describe randomized matrix algorithms for two fundamental matrix problems: Section 4 will be devoted to describing several related algorithms for the least-squares approximation problem; and Section 5 will be devoted to describing several related algorithms for the problem of low-rank matrix approximation. Then, Section 6 will describe in more detail some of these issues from

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

an empirical perspective, with an emphasis on the ways that statistical leverage scores have been used more generally in large-scale data analysis; Section 7 will provide some more general thought on this successful technology transfer experience; and Section 8 will provide a brief conclusion.

2

Matrices in Large-scale Scientific Data Analysis

In this section, we will provide a brief overview of examples of applications of randomized matrix algorithms in large-scale scientific data analysis. Although far from exhaustive, these examples should serve as a motivation to illustrate several complementary perspectives that one can adopt on these techniques.

2.1 A Brief Background

Matrices arise in machine learning and modern massive data set (MMDS) analysis in many guises. One broad class of matrices to which randomized algorithms have been applied is *object-feature matrices.*

- **Matrices from object-feature data.** An $m \times n$ real-valued matrix A provides a natural structure for encoding information about m objects, each of which is described by n features. In astronomy, for example, very small angular regions of the sky imaged at a range of electromagnetic frequency bands can be represented as a matrix — in that case, an object is a region and the features are the elements of the frequency bands. Similarly, in genetics, DNA

> Single Nucleotide Polymorphism or DNA microarray expression data can be represented in such a framework, with A_{ij} representing the expression level of the i-th gene or SNP in the j-th experimental condition or individual. Similarly, term-document matrices can be constructed in many Internet applications, with A_{ij} indicating the frequency of the j-th term in the i-th document.

Matrices arise in many other contexts — e.g., they arise when solving partial differential equations in scientific computation as discretizations of continuum operators; and they arise as so-called kernels when describing pairwise relationships between data points in machine learning. In many of these cases, certain conditions — e.g., that the spectrum decays fairly quickly or that the matrix is structured such that it can be applied quickly to arbitrary vectors or that the elements of the matrix satisfy some smoothness conditions — are known or are thought to hold.

A fundamental property of matrices that is of broad applicability in both data analysis and scientific computing is the Singular Value Decomposition (SVD). If $A \in \mathbb{R}^{m\times n}$, then there exist orthogonal matrices $U = [u^1 u^2 \dots u^m] \in \mathbb{R}^{m\times m}$ and $V = [v^1 v^2 \dots v^n] \in \mathbb{R}^{n\times n}$ such that $U^T A V = \Sigma = \mathbf{diag}(\sigma_1, \dots, \sigma_\nu)$, where $\Sigma \in \mathbb{R}^{m\times n}$, $\nu = \min\{m,n\}$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_\nu \geq 0$. The σ_i are the singular values of A, the column vectors u^i, v^i are the i-th left and the i-th right singular vectors of A, respectively. If $k \leq r = \text{rank}(A)$, then the SVD of A may be written as

$$\begin{aligned} A = U\Sigma V^T &= [U_k \; U_{k,\perp}] \begin{bmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0} & \Sigma_{k,\perp} \end{bmatrix} \begin{bmatrix} V_k^T \\ V_{k,\perp}^T \end{bmatrix} \\ &= U_k \Sigma_k V_k^T + U_{k,\perp} \Sigma_{k,\perp} V_{k,\perp}^T. \end{aligned} \tag{2.1}$$

Here, Σ_k is the $k \times k$ diagonal matrix containing the top k singular values of A, and $\Sigma_{k,\perp}$ is the $(r-k) \times (r-k)$ diagonal matrix containing the bottom $r-k$ nonzero singular values of A, V_k^T is the $k \times n$ matrix consisting of the corresponding top k right singular vectors,[1] etc.

[1] In the text, we will sometimes overload notation and use V_k^T to refer to *any* $k \times n$ orthonormal matrix spanning the space spanned by the top-k right singular vectors (and similarly

By keeping just the top k singular vectors, the matrix $A_k = U_k \Sigma_k V_k^T$ is the best rank-k approximation to A, when measured with respect to the spectral and Frobenius norm. Let $||A||_F^2 = \sum_{i=1}^m \sum_{j=1}^n A_{ij}^2$ denote the square of the Frobenius norm; let $||A||_2 = \sup_{x \in \mathbb{R}^n,\ x \neq 0} ||Ax||_2 / ||x||_2$ denote the spectral norm;[2] and, 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.

Finally, since they will play an important role in later developments, the *statistical leverage scores* of an $m \times n$ matrix, with $m > n$, are defined here.

Definition 2.1. Given an arbitrary $m \times n$ matrix A, with $m > n$, let U denote the $m \times n$ matrix consisting of the n left singular vectors of A, and let $U_{(i)}$ denote the i-th row of the matrix U as a row vector. Then, the quantities

$$\ell_i = ||U_{(i)}||_2^2, \quad \text{for } i \in \{1, \ldots, m\},$$

are the *statistical leverage scores* of the rows of A.

Several things are worth noting about this definition. First, although we have defined these quantities in terms of a particular basis, they clearly do not depend on that particular basis, but instead 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, $\ell_i = ||U_{(i)}||_2^2 = (UU^T)_{ii} = (P_A)_{ii}$. That is, the statistical leverage scores of a matrix A are equal to the diagonal elements of the projection matrix onto

for U_k and the left singular vectors). The reason is that this basis is used only to compute the importance sampling probabilities — since those probabilities are proportional to the diagonal elements of the projection matrix onto the span of this basis, the particular basis does not matter.

[2] Since the spectral norm is the largest singular value of the matrix, it is an "extremal" norm in that it measures the worst-case stretch of the matrix, while the Frobenius norm is more of an "averaging" norm, since it involves a sum over every singular direction. The former is of greater interest in scientific computing and NLA, where one is interested in actual columns for the subspaces they define and for their good numerical properties, while the latter is of greater interest in data analysis and machine learning, where one is more interested in actual columns for the features they define. Both are of interest in this review.

the span of its columns. Second, if $m > n$, then $O(mn^2)$ 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. Third, one could also define leverage scores for the columns of such a matrix A, but clearly those are all equal to one unless $m < n$ or A is rank-deficient. Fourth, and more generally, given a rank parameter k, one can define the *statistical leverage scores relative to the best rank-k approximation to* A to be the m diagonal elements of the projection matrix onto the span of the columns of A_k, the best rank-k approximation to A. Finally, the *coherence* γ of the rows of A is $\gamma = \max_{i \in \{1,\ldots,m\}} \ell_i$, i.e., it is the largest statistical leverage score of A.

2.2 Motivating Scientific Applications

To illustrate a few examples where randomized matrix algorithms have already been applied in scientific data analysis, recall that "the human genome" consists of a sequence of roughly 3 billion base pairs on 23 pairs of chromosomes, roughly 1.5% of which codes for approximately 20,000–25,000 proteins. A DNA microarray is a device that can be used to measure simultaneously the genome-wide response of the protein product of each of these genes for an individual or group of individuals in numerous different environmental conditions or disease states. This very coarse measure can, of course, hide the individual differences or polymorphic variations. There are numerous types of polymorphic variation, but the most amenable to large-scale applications is the analysis of Single Nucleotide Polymorphisms (SNPs), which are known locations in the human genome where two alternate nucleotide bases (or alleles, out of A, C, G, and T) are observed in a non-negligible fraction of the population. These SNPs occur quite frequently, roughly 1 base pair per thousand (depending on the minor allele frequency), and thus they are effective genomic markers for the tracking of disease genes (i.e., they can be used to perform classification into sick and not sick) as well as population histories (i.e., they can be used to infer properties about population genetics and human evolutionary history).

In both cases, $m \times n$ matrices A naturally arise, either as a people-by-gene matrix, in which A_{ij} encodes information about the response of the j^{th} gene in the i^{th} individual/condition, or as people-by-SNP matrices, in which A_{ij} encodes information about the value of the j^{th} SNP in the i^{th} individual. Thus, matrix computations have received attention in these genetics applications [8, 112, 125, 131, 145, 165]. To give a rough sense of the sizes involved, if the matrix is constructed in the naïve way based on data from the International HapMap Project [185, 186], then it is of size roughly 400 people by 10^6 SNPs, although more recent technological developments have increased the number of SNPs to well into the millions and the number of people into the thousands and tens-of-thousands. Depending on the size of the data and the genetics problem under consideration, randomized algorithms can be useful in one or more of several ways.

For example, a common genetics challenge is to determine whether there is any evidence that the samples in the data are from a population that is structured, i.e., are the individuals from a homogeneous population or from a population containing genetically distinct subgroups? In medical genetics, this arises in case-control studies, where uncorrected population structure can induce false positives; and in population genetics, it arises where understanding the structure is important for uncovering the demographic history of the population under study. To address this question, it is common to perform a procedure such as the following. Given an appropriately-normalized (where, of course, the normalization depends crucially on domain-specific considerations) $m \times n$ matrix A:

- Compute a full or partial SVD or perform a QR decomposition, thereby computing the eigenvectors and eigenvalues of the correlation matrix AA^T.
- Appeal to a statistical model selection criterion[3] to determine either the number k of principal components to keep in order

[3] For example, the model selection rule could compare the top part of the spectrum of the data matrix to that of a random matrix of the same size [164, 91]; or it could use the full spectrum to compute a test statistic to determine whether there is more structure in the data matrix than would be present in a random matrix of the same size [165, 116].

to project the data onto or whether to keep an additional principal component as significant.

Although this procedure could be applied to any data set A, to obtain meaningful genetic conclusions one must deal with numerous issues.[4] In any case, however, the computational bottleneck is typically computing the SVD or a QR decomposition. For small to medium-sized data, this is not a problem — simply call MATLAB or call appropriate routines from LAPACK directly. The computation of the full eigendecomposition takes $O(\min\{mn^2, m^2n\})$ time, and if only k components of the eigendecomposition are needed then the running time is typically $O(mnk)$ time. (This "typically" is awkward from the perspective of worst-case analysis, but it is not usually a problem in practice. Of course, one could compute the full SVD in $O(\min\{mn^2, m^2n\})$ time and truncate to obtain the partial SVD. Alternatively, one could use a Krylov subspace method to compute the partial SVD in $O(mnk)$ time, but these methods can be less robust. Alternatively, one could perform a rank-revealing QR factorization such as that of Gu and Eisenstat [105] and then post-process the factors to obtain a partial SVD. The cost of computing the QR decomposition is typically $O(mnk)$ time, although these methods can require slightly longer time in rare cases [105]. See [107] for a discussion of these topics.)

Thus, these traditional methods can be quite fast even for very large data if one of the dimensions is small, e.g., 10^2 individuals typed at 10^7 SNPs. On the other hand, if both m and n are large, e.g., 10^3 individuals at 10^6 SNPs, or 10^4 individuals at 10^5 SNPs, then, for interesting values of the rank parameter k, the $O(mnk)$ running time of even the QR decomposition can be prohibitive. As we will see below, however, by exploiting randomness inside the algorithm, one can obtain an $O(mn \log k)$ running time. (All of this assumes that the data matrix is dense and fits in memory, as is typical in SNP applications. More generally, randomized matrix algorithms to be reviewed below also help in other computational environments, e.g., for sparse input matrices, for

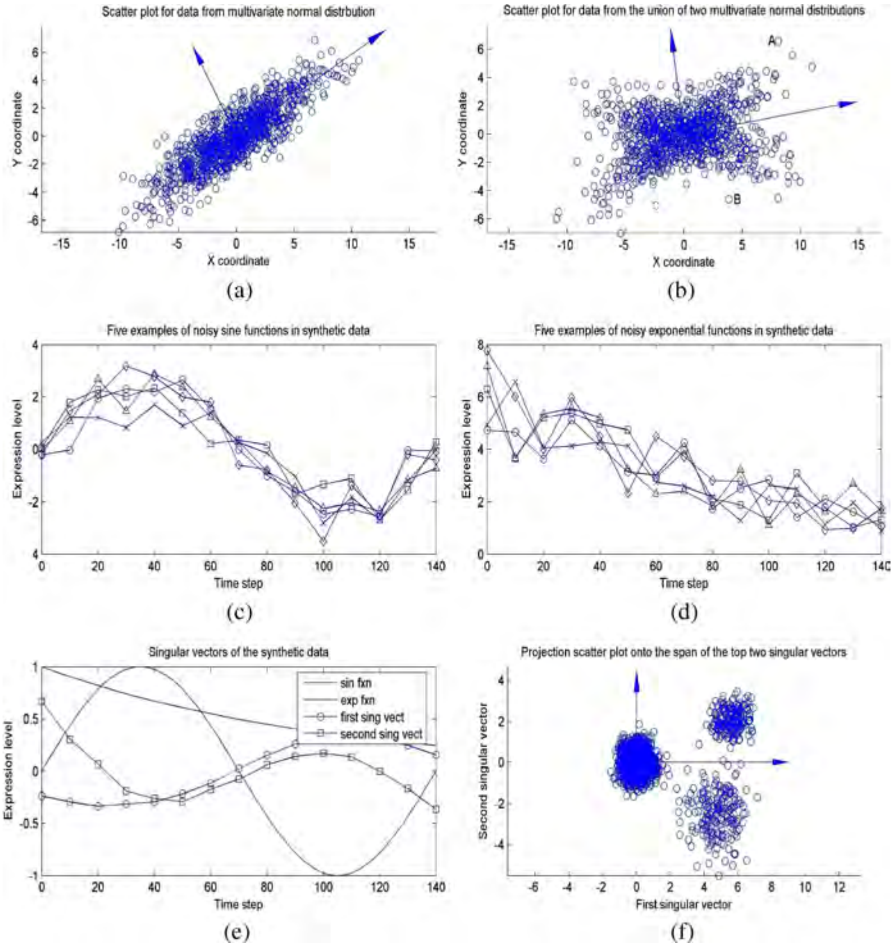
[4] For example, how to normalize the data, how to deal with missing data, how to correct for linkage disequilibrium (or correlational) structure in the genome, how to correct for closely-related individuals within the sample, etc.

matrices too large to fit into main memory, when one wants to reorganize the steps of the computations to exploit modern multi-processor architectures, etc. See [107] for a discussion of these topics.) Since interesting values for k are often in the hundreds, this improvement from $O(k)$ to $O(\log k)$ can be quite significant in practice; and thus one can apply the above procedure for identifying structure in DNA SNP data on much larger data sets than would have been possible with traditional deterministic methods [93].

More generally, a common *modus operandi* in applying NLA and matrix techniques such as PCA and the SVD to DNA microarray, DNA SNPs, and other data problems is:

- Model the people-by-gene or people-by-SNP data as an $m \times n$ matrix A.
- Perform the SVD (or related eigen-methods such as PCA or recently-popular manifold-based methods [170, 175, 184] that boil down to the SVD, in that they perform the SVD or an eigendecomposition on nontrivial matrices constructed from the data) to compute a small number of eigengenes or eigenSNPs or eigenpeople that capture most of the information in the data matrix.
- Interpret the top eigenvectors as meaningful in terms of underlying biological processes; or apply a heuristic to obtain actual genes or actual SNPs from the corresponding eigenvectors in order to obtain such an interpretation.

In certain cases, such reification may lead to insight and such heuristics may be justified. For instance, if the data happen to be drawn from a Guassian distribution, as in Figure 2.1(a), then the eigendirections tend to correspond to the axes of the corresponding ellipsoid, and there are many vectors that, up to noise, point along those directions. In most cases, however, e.g., when the data are drawn from the union of two normals (or mixture of two Gaussians), as in Figure 2.1(b), such reification is not valid. In general, the justification for interpretation comes from domain knowledge and not the mathematics [101, 125, 146, 137]. The reason is that the eigenvectors themselves, being mathematically defined abstractions, can be calculated for any data matrix and thus



(a)

(b)

(c)

(d)

(e)

(f)

Fig. 2.1. Applying the SVD to data matrices A. (a) 1000 points on the plane, corresponding to a 1000×2 matrix A, (and the two principal components) drawn from a multivariate normal distribution. (b) 1000 points on the plane (and the two principal components) drawn from a more complex distribution, in this case the union of two multivariate normal distributions. (c–f) A synthetic data set considered in [191] to model oscillatory and exponentially decaying patterns of gene expression from [52], as described in the text. (c) Overlays of five noisy sine wave genes. (d) Overlays of five noisy exponential genes. (e) The first and second singular vectors of the data matrix (which account for 64% of the variance in the data), along with the original sine pattern and exponential pattern that generated the data. (f) Projection of the synthetic data on its top two singular vectors. Although the data cluster well in the low-dimensional space, the top two singular vectors are completely artificial and do not offer insight into the oscillatory and exponentially decaying patterns that generated the data.

are not easily understandable in terms of processes generating the data: eigenSNPs (being linear combinations of SNPs) cannot be assayed; nor can eigengenes (being linear combinations of genes) be isolated and purified; nor is one typically interested in how eigenpatients (being linear combinations of patients) respond to treatment when one visits a physician.

For this and other reasons, a common task in genetics and other areas of data analysis is the following: given an input data matrix A and a parameter k, find the best subset of exactly k *actual* DNA SNPs or *actual* genes, i.e., *actual* columns or rows from A, to use to cluster individuals, reconstruct biochemical pathways, reconstruct signal, perform classification or inference, etc. Unfortunately, common formalizations of this algorithmic problem — including looking for the k actual columns that capture the largest amount of information or variance in the data or that are maximally uncorrelated — lead to intractable optimization problems [53, 54]. For example, consider the so-called Column Subset Selection Problem [33]: given as input an arbitrary $m \times n$ matrix A and a rank parameter k, choose the set of exactly k columns of A s.t. the $m \times k$ matrix C minimizes (over all $\binom{n}{k}$ sets of such columns) the error:

$$\min ||A - P_C A||_\nu = \min ||A - CC^+ A||_\nu \tag{2.2}$$

where $\nu \in \{2, F\}$ represents the spectral or Frobenius norm of A, C^+ is the Moore-Penrose pseudoinverse of C, and $P_C = CC^+$ is the projection onto the subspace spanned by the columns of C. As we will see below, however, by exploiting randomness inside the algorithm, one can find a small set of actual columns that is provably nearly optimal. Moreover, this algorithm and obvious heuristics motivated by it have already been applied successfully to problems of interest to geneticists such as genotype reconstruction in unassayed populations, identifying substructure in heterogeneous populations, and inference of individual ancestry [72, 114, 137, 161, 162, 163, 164].

In order to understand better the reification issues in scientific data analysis, consider a synthetic data set — it was originally introduced in [191] to model oscillatory and exponentially decaying patterns of gene expression from [52], although it could just as easily be used to describe oscillatory and exponentially decaying patterns in stellar spectra, etc. The data matrix consists of 14 expression level assays (columns of A) and 2000 genes (rows of A), corresponding to a 2000×14 matrix A. Genes have one of three types of transcriptional response: noise (1600 genes); noisy sine pattern (200 genes); and noisy exponential pattern (200 genes). Figures 2.1(c) and 2.1d present the "biological" data, i.e., overlays of five noisy sine wave genes and five noisy exponential genes, respectively; Figure 2.1(e) presents the first and second singular vectors of the data matrix, along with the original sine pattern and exponential pattern that generated the data; and Figure 2.1(f) shows that the data cluster well in the space spanned by the top two singular vectors, which in this case account for 64% of the variance in the data. Note, though, that the top two singular vectors both display a linear combination of oscillatory and decaying properties; and thus they are not easily interpretable as "latent factors" or "fundamental modes" of the original (sinusoid and exponential) "biological" processes generating the data. This is problematic more generally when one is interested in extracting insight or "discovering knowledge" from the output of data analysis algorithms [137].[5]

Broadly similar issues arise in many other MMDS (modern massive data sets) application areas. In astronomy, for example, PCA and the SVD have been used directly for spectral classification [57, 58, 133, 194], to predict morphological types using galaxy spectra [85], to select quasar candidates from sky surveys [195], etc. [12, 29, 37, 143]. Size is an issue, but so too is understanding the data [11, 36]; and many of these studies have found that principal components of galaxy spectra (and their elements) correlate with various physical processes such as star formation (via absorption and emission line strengths of, e.g., the so-called Hα spectral line) as well as with galaxy color and morphology. In addition, there are many applications in scientific computing where

[5] Indeed, after describing the many uses of the vectors provided by the SVD and PCA in DNA microarray analysis, Kuruvilla et al. [125] bluntly conclude that "While very efficient basis vectors, the (singular) vectors themselves are completely artificial and do not correspond to actual (DNA expression) profiles. ... Thus, it would be interesting to try to find basis vectors for all experiment vectors, using actual experiment vectors and not artificial bases that offer little insight."

low-rank matrices appear, e.g., fast numerical algorithms for solving partial differential equations and evaluating potential fields rely on low-rank approximation of continuum operators [104, 102], and techniques for model reduction or coarse graining of multiscale physical models that involve rapidly oscillating coefficients often employ low-rank linear mappings [78]. Recent work that has already used randomized low-rank matrix approximations based on those reviewed here include [39, 42, 79, 129, 130, 140]. More generally, many of the machine learning and data analysis applications cited below use these algorithms and/or greedy or heuristic variants of these algorithms for problems in diagnostic data analysis and for unsupervised feature selection for classification and clustering problems.

2.3 Randomization as a Resource

The examples described in the previous subsection illustrate two common reasons for using randomization in the design of matrix algorithms for large-scale data problems:

- **Faster Algorithms.** In some computation-bound applications, one simply wants *faster algorithms* that return more-or-less the exact[6] answer. In many of these applications, one thinks of the rank parameter k as the numerical rank of the matrix,[7] and thus one wants to choose the error parameter ϵ such that the approximation is precise on the order of machine precision.
- **Interpretable Algorithms.** In other analyst-bound applications, one wants *simpler algorithms or more-interpretable output* in order to obtain qualitative insight in order to pass

[6] Say, for example, that a numerically-stable answer that is precise to, say, 10 digits of significance is more-or-less exact. Exact answers are often impossible to compute numerically, in particular for continuous problems, as anyone who has studied numerical analysis knows. Although they will not be the main focus of this review, such issues need to be addressed to provide high-quality numerical implementations of the randomized algorithms discussed here.

[7] Think of the numerical rank of a matrix as being its “true” rank, up to issues associated with machine precision and roundoff error. Depending on the application, it can be defined in one of several related ways. For example, if $\nu = \min\{m,n\}$, then, given a tolerance parameter ε, one way to define it is the largest k such that $\sigma_{\nu-k+1} > \varepsilon \cdot \sigma_\nu$ [98].

> to a downstream analyst.[8] In these cases, k is determined according to some domain-determined model selection criterion, in which case the difference between σ_k and σ_{k+1} may be small or it may be that $\sigma_{k+1} \gg 0$.[9] Thus, it is acceptable (or even desirable since there is substantial noise in the data) if ϵ is chosen such that the approximation is much less precise.

Thus, randomization can be viewed as a computational resource to be exploited in order to lead to "better" algorithms. Perhaps the most obvious sense of better is faster running time, either in worst-case asymptotic theory and/or numerical implementation — we will see below that both of these can be achieved. But another sense of better is that the algorithm is more useful or easier to use — e.g., it may lead to more interpretable output, which is of interest in many data analysis applications where analyst time rather than just computational time is of interest. Of course, there are other senses of better — e.g., the use of randomization and approximate computation can lead implicitly to regularization and more robust output; randomized algorithms can be organized to exploit modern computational architectures better than classical numerical methods; and the use of randomization can lead to simpler algorithms that are easier to analyze or reason about when applied in counterintuitive settings[10] — but these will not be the main focus of this review.

[8] The tension between providing more interpretable decompositions versus optimizing any single criterion — say, obtaining slightly better running time (in scientific computing) or slightly better prediction accuracy (in machine learning) — is well-known [137]. It was illustrated most prominently recently by the Netflix Prize competition — whereas a half dozen or so base models captured the main ideas, the winning model was an ensemble of over 700 base models [121].

[9] Recall that σ_i is the i^{th} singular value of the data matrix.

[10] Randomization can also be useful in less obvious ways — e.g., to deal with pivot rule issues in communication-constrained linear algebra [10], or to achieve improved condition number properties in optimization applications [62].

3

Randomization Applied to Matrix Problems

Before describing recently-developed randomized algorithms for least-squares approximation and low-rank matrix approximation that underlie applications such as those described in Section 2, in this section we will provide a brief overview of the immediate precursors[1] of that work.

3.1 Random Sampling and Random Projections

Given an $m \times n$ matrix A, it is often of interest to sample randomly a small number of actual columns from that matrix.[2] (To understand

[1] Although this "first-generation" of randomized matrix algorithms was extremely fast and came with provable quality-of-approximation guarantees, most of these algorithms did *not* gain a foothold in NLA and only heuristic variants of them were used in machine learning and data analysis applications. Understanding them, though, was important in the development of a "second-generation" of randomized matrix algorithms that were embraced by those communities. For example, while in some cases these first-generation algorithms yield to a more sophisticated analysis and thus can be implemented directly, more often these first-generation algorithms represent a set of primitives that are more powerful once the randomness is decoupled from the linear algebraic structure.

[2] Alternatively, one might be interested in sampling other things like elements [2] or submatrices [89]. Like the algorithms described in this section, these other sampling algorithms also achieve additive-error bounds. We will not describe them in this review since, although of interest in TCS, they have not (yet?) gained traction in either NLA or in machine learning and data analysis applications.

why sampling columns (or rows) from a matrix is of interest, recall that matrices are "about" their columns and rows [180] — that is, linear combinations are taken with respect to them; one all but understands a given matrix if one understands its column space, row space, and null spaces; and understanding the subspace structure of a matrix sheds a great deal of light on the linear transformation that the matrix represents.) A naïve way to perform this random sampling would be to select those columns uniformly at random in i.i.d. trials. A more sophisticated and much more powerful way to do this would be to construct an *importance sampling distribution* $\{p_i\}_{i=1}^n$, and then perform the random sample according to it.

To illustrate this importance sampling approach in a simple setting, consider the problem of approximating the product of two matrices. Given as input any arbitrary $m \times n$ matrix A and any arbitrary $n \times p$ matrix B:

- Compute the importance sampling probabilities $\{p_i\}_{i=1}^n$, where
$$p_i = \frac{||A^{(i)}||_2 ||B_{(i)}||_2}{\sum_{i'=1}^n ||A^{(i')}||_2 ||B_{(i')}||_2}. \tag{3.1}$$
- Randomly select (and rescale appropriately — if the j^{th} column of A is chosen, then scale it by $1/\sqrt{cp_j}$; see [69] for details) c columns of A and the corresponding rows of B (again rescaling in the same manner), thereby forming $m \times c$ and $c \times p$ matrices C and R, respectively.

Two quick points are in order regarding the sampling process in this and other randomized algorithms to be described below. First, the sampling here is with replacement. Thus, in particular, if $c = n$ one does not necessarily recover the "exact" answer, but of course one should think of this algorithm as being most appropriate when $c \ll n$. Second, if a given column-row pair is sampled, then it must be rescaled by a factor depending on the total number of samples to be drawn and the probability that given column-row pair was chosen. The particular form of $1/cp_j$ ensures that appropriate estimators are unbiased; see [69] for details.

This algorithm (as well as other algorithms that sample based on the Euclidean norms of the input matrices) requires just two passes over the data from external storage. Thus, it can be implemented in pass-efficient [69] or streaming [151] models of computation. This algorithm is described in more detail in [69], where it is shown that Frobenius norm bounds of the form

$$||AB - CR||_F \leq \frac{O(1)}{\sqrt{c}} ||A||_F ||B||_F, \tag{3.2}$$

where $O(1)$ refers to some constant, hold both in expectation and with high probability. (Issues associated with potential failure probabilities, big-O notation, etc. for this pedagogical example are addressed in [69] — these issues will be addressed in more detail for the algorithms of the subsequent sections.) Moreover, if, instead of using importance sampling probabilities of the form (3.1) that depend on both A and B, one uses probabilities of the form

$$p_i = ||A^{(i)}||_2^2 / ||A||_F^2 \tag{3.3}$$

that depend on only A (or alternatively ones that depend only on B), then (slightly weaker) bounds of the form (3.2) still hold [69]. As we will see, this algorithm (or variants of it, as well as their associated bounds) is a primitive that underlies many of the randomized matrix algorithms that have been developed in recent years; for very recent examples of this, see [135, 81].

To gain insight into "why" this algorithm works, recall that the product AB may be written as the outer product or sum of n rank one matrices $AB = \sum_{t=1}^{n} A^{(t)} B_{(t)}$. When matrix multiplication is formulated in this manner, a simple randomized algorithm to approximate the product matrix AB suggests itself: randomly sample with replacement from the terms in the summation c times, rescale each term appropriately, and output the sum of the scaled terms. If $m = p = 1$ then $A^{(t)}, B_{(t)} \in \mathbb{R}$ and it is straightforward to show that this sampling procedure produces an unbiased estimator for the sum. When the terms in the sum are rank one matrices, similar results hold. In either case, using importance sampling probabilities to exploit non-uniformity structure in the data — e.g., to bias the sample toward "larger" terms in the sum, as (3.1) does — produces an estimate with *much* better variance properties. For example, importance sampling probabilities of the form (3.1) are optimal with respect to minimizing the expectation of $||AB - CR||_F$.

The analysis of the Frobenius norm bound (3.2) is quite simple [69], using very simple linear algebra and only elementary probability, and it can be improved. Most relevant for the randomized matrix algorithms of this review is the bound of [171, 172], where much more sophisticated methods were used to shown that if $B = A^T$ is an $n \times k$ orthogonal matrix Q (i.e., its k columns consist of k orthonormal vectors in $\mathbb{R}^n$),[3] then, under certain assumptions satisfied by orthogonal matrices, spectral norm bounds of the form

$$\left\|I - CC^T\right\|_2 = \left\|Q^TQ - Q^TSS^TQ\right\|_2 \leq O(1)\sqrt{\frac{k \log c}{c}} \tag{3.4}$$

hold both in expectation and with high probability. In this and other cases below, one can represent the random sampling operation with a *random sampling matrix* S — e.g., if the random sampling is implemented by choosing c columns, one in each of c i.i.d. trials, then the $n \times c$ matrix S has entries $S_{ij} = 1/\sqrt{cp_i}$ if the i^{th} column is picked in the j^{th} independent trial, and $S_{ij} = 0$ otherwise — in which case $C = AS$.

Alternatively, given an $m \times n$ matrix A, one might be interested in performing a random projection by post-multiplying A by an $n \times \ell$ *random projection matrix* Ω, thereby selecting ℓ linear combinations of the columns of A. There are several ways to construct such a matrix.

- Johnson and Lindenstrauss consider an orthogonal projection onto a random ℓ-dimensional space [115], where $\ell = O(\log m)$, and [88] considers a projection onto ℓ random orthogonal vectors. (In both cases, as well as below, the obvious scaling factor of $\sqrt{n/\ell}$ is needed.)

[3] In this case, $Q^TQ = I_k$, $\|Q\|_2 = 1$, and $\|Q\|_F^2 = k$. Thus, the right hand side of (3.2) would be $O(1)\sqrt{k^2/c}$. The tighter spectral norm bounds of the form (3.4) on the approximate product of two orthogonal matrices can be used to show that all the singular values of Q^TS are nonzero and thus that rank is not lost — a crucial step in relative-error and high-precision randomized matrix algorithms.

- [113] and [61] choose the entries of Ω as independent, spherically-symmetric random vectors, the coordinates of which are ℓ i.i.d. Gaussian $N(0,1)$ random variables.
- [1] chooses the entries of $n \times \ell$ matrix Ω as $\{-1,+1\}$ random variables and also shows that a constant factor — up to $2/3$ — of the entries of Ω can be set to 0.
- [3, 4, 142] choose $\Omega = DHP$, where D is a $n \times n$ diagonal matrix, where each D_{ii} is drawn independently from $\{-1,+1\}$ with probability $1/2$; H is an $n \times n$ normalized Hadamard transform matrix, defined below; and P is an $n \times \ell$ random matrix constructed as follows: $P_{ij} = 0$ with probability $1-q$, where $q = O(\log^2(m)/n)$; and otherwise either P_{ij} is drawn from a Gaussian distribution with an appropriate variance, or P_{ij} is drawn independently from $\big\{-\sqrt{1/\ell q}, +\sqrt{1/\ell q}\big\}$, each with probability $q/2$.

As with random sampling matrices, post-multiplication by the $n \times \ell$ random projection matrix Ω amounts to operating on the columns — in this case, choosing linear combinations of columns; and thus pre-multiplying by Ω^T amounts to choosing a small number of linear combinations of rows. Note that, aside from the original constructions, these randomized linear mappings are *not* random projections in the usual linear algebraic sense; but instead they satisfy certain approximate metric preserving properties satisfied by "true" random projections, and they are useful much more generally in algorithm design. Vanilla application of such random projections has been used in data analysis and machine learning applications for clustering and classification of data [24, 84, 87, 95, 120, 190].

An important technical point is that the last Hadamard-based construction is of particular importance for fast implementations (both in theory and in practice). Recall that the (non-normalized) $n \times n$ matrix of the Hadamard transform H_n may be defined recursively as

$$H_n = \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix}, \quad \text{with} \quad H_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix},$$

in which case the $n \times n$ normalized matrix of the Hadamard transform, to be denoted by H hereafter, is equal to $\frac{1}{\sqrt{n}} H_n$. (For readers not familiar with the Hadamard transform, note that it is an orthogonal transformation, and one should think of it as a real-valued version of the complex Fourier transform. Also, as defined here, n is a power of 2, but variants of this construction exist for other values of n.) Importantly, applying the *randomized Hadamard transform*, i.e., computing the product xDH for any vector $x \in \mathbb{R}^n$ takes $O(n \log n)$ time (or even $O(n \log r)$ time if only r elements in the transformed vector need to be accessed). Applying such a *structured random projection* was first proposed in [3, 4], it was first applied in the context of randomized matrix algorithms in [77, 174], and there has been a great deal of research in recent years on variants of this basic structured random projection that are better in theory or in practice [5, 6, 7, 9, 60, 77, 118, 119, 127, 128, 142, 169]. For example, one could choose $\Omega = DHS$, where S is a random sampling matrix, as defined above, that represents the operation of uniformly sampling a small number of columns from the randomized Hadamard transform.

Random projection matrices constructed with any of these methods exhibit a number of similarities, and the choice of which is appropriate depends on the application — e.g., a random unitary matrix or a matrix with i.i.d. Gaussian entries may be the simplest conceptually or provide the strongest bounds; for TCS algorithmic applications, one may prefer a construction with i.i.d. Gaussian, $\{-1,+1\}$, etc. entries, or randomized Hadamard methods that are theoretically efficient to implement; for numerical implementations, one may prefer i.i.d. Gaussians if one is working with structured matrices that can be applied rapidly to arbitrary vectors and/or if one wants to be very aggressive in minimizing the oversampling factor needed by the algorithm, while one may prefer fast-Fourier-based methods that are better by constant factors than simple Hadamard-based constructions when working with arbitrary dense matrices.

Intuitively, these random projection algorithms work since, if $\Omega^{(j)}$ is the j^{th} column of Ω, then $A\Omega^{(j)}$ is a random vector in the range of A. Thus if one generates several such vectors, they will be linearly-independent (with very high probability, but perhaps poorly conditioned), and so one might hope to get a good approximation to the best rank-k approximation to A by choosing k or slightly more than k such vectors. Somewhat more technically, one can prove that these random projection algorithms work by establishing variants of the basic *Johnson-Lindenstrauss (JL) lemma*, which states:

- Any set of n points in a high-dimensional Euclidean space can be embedded (via the constructed random projection) into an ℓ-dimensional Euclidean space, where ℓ is logarithmic in n and independent of the ambient dimension, such that all the pairwise distances are preserved to within an arbitrarily-small multiplicative (or $1 \pm \epsilon$) factor [1, 3, 4, 61, 88, 113, 115, 142].

This result can then be applied to $\binom{n}{2}$ vectors associated with the columns of A. The most obvious (but not necessarily the best) such set of vectors is the rows of the original matrix A, in which case one shows that the random variable $||A_{(i)}\Omega - A_{(i')}\Omega||_2^2$ equals $||A_{(i)} - A_{(i')}||_2^2$ in expectation (which is usually easy to show) and that the variance is sufficiently small (which is usually harder to show).

By now, the relationship between sampling algorithms and projection algorithms should be clear. Random sampling algorithms identify a coordinate-based non-uniformity structure, and they use it to construct an importance sampling distribution. For these algorithms, the "bad" case is when that distribution is extremely nonuniform, i.e., when most of the probability mass is localized on a small number of columns. This is the bad case for sampling algorithms in the sense that a naïve method like uniform sampling will perform poorly, while using an importance sampling distribution that provides a bias toward these columns will perform much better (at preserving distances, angles, subspaces, and other quantities of interest). On the other hand, random projections and randomized Hadamard transforms destroy or "wash out" or uniformize that coordinate-based non-uniformity structure by rotating to a basis where the importance sampling distribution is very delocalized and thus where uniform sampling is nearly optimal (but by satisfying the above JL lemma they too preserve metric properties of interest). For readers more familiar with Dirac δ functions and sinusoidal functions, recall that a similar situation holds — δ functions are extremely localized, but when they are multiplied by a Fourier transform, they are converted into delocalized sinusoids. As we will see, making such structure explicit has numerous benefits.

3.2 Randomization for Large-scale Matrix Problems

Consider the following random projection algorithm that was introduced in the context of understanding the success of latent semantic analysis [159]. Given an $m \times n$ matrix A and a rank parameter k:

- Construct an $n \times \ell$, with $\ell \geq \alpha \log m/\epsilon^2$ for some constant α, random projection matrix Ω, as in the previous subsection.
- Return $B = A\Omega$.

This algorithm, which amounts to choosing uniformly a small number ℓ of columns in a randomly rotated basis, was introduced in [159], where it is proven that

$$||A - P_{B_{2k}}A||_F \leq ||A - P_{U_k}A||_F + \epsilon||A||_F \tag{3.5}$$

holds with high probability. (Here, B_{2k} is the best rank-$2k$ approximation to the matrix B; $P_{B_{2k}}$ is the projection matrix onto this $2k$-dimensional space; and P_{U_k} is the projection matrix onto U_k, the top k left singular vectors of A.) The analysis of this algorithm boils down to the JL ideas of the previous subsection applied to the rows of the input matrix A. That is, the error $||A - P_{B_{2k}}A||_F$ boils down to the error incurred by the best rank-$2k$ approximation plus an additional error term. By applying the relative-error JL lemma to the rows of the matrix A, the additional error can be shown to be no greater than $\epsilon\,\|A\|_F$.

Next, consider the following random sampling algorithm that was introduced in the context of clustering large data sets [68]. Given an $m \times n$ matrix A and a rank parameter k:

- Compute the importance sampling probabilities $\{p_i\}_{i=1}^n$, where $p_i = ||A^{(i)}||_2^2/||A||_F^2$.
- Randomly select and rescale $c = O(k \log k/\epsilon^2)$ columns of A according to these probabilities to form the matrix C.

This algorithm was introduced in [68], although a more complex variant of it appeared in [90]. The original analysis was extended and simplified in [70], where it is proven that

$$\|A - P_{C_k}A\|_2 \leq \|A - P_{U_k}A\|_2 + \epsilon\|A\|_F \quad \text{and} \tag{3.6}$$

$$\|A - P_{C_k}A\|_F \leq \|A - P_{U_k}A\|_F + \epsilon\|A\|_F \tag{3.7}$$

hold with high probability. (Here, C_k is the best rank-k approximation to the matrix C, and P_{C_k} is the projection matrix onto this k-dimensional space.) This additive-error column-based matrix decomposition, as well as heuristic variants of it, has been applied in a range of data analysis applications [139, 163, 181, 188, 158].

Note that, in a theoretical sense, this and related random sampling algorithms that sample with respect to the Euclidean norms of the input columns are particularly appropriate for very large-scale settings. The reason is that these algorithms can be implemented efficiently in the pass-efficient or streaming models of computation, in which the scarce computational resources are the number of passes over the data, the additional RAM space required, and the additional time required. See [69, 70] for details about this.

The analysis of this random sampling algorithm boils down to an approximate matrix multiplication result, in a sense that will be constructive to consider in some detail. As an intermediate step in the proof of the previous results, that was made explicit in [70], it was shown that

$$\|A - P_{C_k}A\|_2^2 \leq ||A - P_{U_k}A||_2^2 + 2\left\|AA^T - CC^T\right\|_2 \quad \text{and}$$

$$\|A - P_{C_k}A\|_F^2 \leq ||A - P_{U_k}A||_F^2 + 2\sqrt{k}\left\|AA^T - CC^T\right\|_F.$$

These bounds decouple the linear algebra from the randomization in the following sense: they hold for *any* set of columns, i.e., for any matrix C, and the effect of the randomization enters only through the “additional error” term. By using $p_i = ||A^{(i)}||_2^2/||A||_F^2$ as the importance sampling probabilities, this algorithm is effectively saying that the relevant non-uniformity structure in the data is defined by the Euclidean norms of the original matrix. (This may be thought to be an appropriate non-uniformity structure to identify since, e.g., it is one that can be identified and extracted in two passes over the data from external storage.) In doing so, this algorithm can take advantage of (3.2) to provide additive-error bounds. A similar thing was seen in the analysis of the random projection algorithm — since the JL lemma was applied directly to the columns of A, additive-error bounds of the form (3.5) were obtained.

This is an appropriate point to pause to describe different notions of approximation that a matrix algorithm might provide. In the theory of algorithms, bounds of the form provided by (3.6) and (3.7) are known as *additive-error bounds*, the reason being that the "additional" error (above and beyond that incurred by the SVD) is an additive factor of the form ϵ times the scale $\|A\|_F$. Bounds of this form are very different and in general weaker than when the additional error enters as a multiplicative factor, such as when the error bounds are of the form $||A - P_{C_k}A|| \leq f(m,n,k,\eta)||A - P_{U_k}A||$, where $f(\cdot)$ is some function and η represents other parameters of the problem. Bounds of this type are of greatest interest when $f(\cdot)$ does not depend on m or n, in which case they are known as a *constant-factor bounds*, or when they depend on m and n only weakly. The strongest bounds are when $f = 1 + \epsilon$, for an error parameter ϵ, i.e., when the bounds are of the form $||A - P_{C_k}A|| \leq (1 + \epsilon)||A - P_{U_k}A||$. These *relative-error bounds* are the gold standard, and they provide a *much* stronger notion of approximation than additive-error or weaker multiplicative-error bounds. We will see bounds of all of these forms below.

One application of these random sampling ideas that deserves special mention is when the input matrix A is symmetric positive semi-definite. Such matrices are common in kernel-based machine learning, and sampling columns in this context often goes by the name *the Nyström method*. Originating in integral equation theory, the Nyström method was introduced into machine learning in [192] and it was analyzed and discussed in detail in [74]. Applications to large-scale machine learning problems include [123, 124, 182] and [126, 197, 198], and applications in statistics and signal processing include [17, 18, 19, 20, 21, 160, 177]. As an example, the Nyström method can be used to provide an approximation to a matrix without even looking at the entire matrix — under assumptions on the input matrix, of course, such as that the leverage scores are approximately uniform.

3.3 A Retrospective and a Prospective

Much of the early work in TCS focused on randomly sampling columns according to an importance sampling distribution that depended on the Euclidean norm of those columns [68, 69, 70, 71, 90, 172]. This had the advantage of being "fast," in the sense that it could be performed in a small number of "passes" over that data from external storage, and also that additive-error quality-of-approximation bounds could be proved. On the other hand, this had the disadvantage of being less immediately-applicable to scientific computing and large-scale data analysis applications. At root, the reason is that these algorithms didn't highlight "interesting" or "relevant" non-uniformity structure, which then led to bounds that were rather coarse. For example, columns are easy to normalize and are often normalized during data preprocessing. Even when not normalized, column norms can still be uninformative, as in heavy-tailed graph data,[4] where they often correlate strongly with simpler statistics such as node degree.

Relatedly, bounds of the form (3.2) do not exploit the underlying vector space structure. This is analogous to how the JL lemma was applied in the analysis of the random projection algorithm — by applying the JL lemma to the actual rows of A, as opposed to some other more refined vectors associated with the rows of A, the underlying vector space structure was ignored and only coarse additive-error bounds were obtained. To obtain improvements and to bridge the gap between TCS, NLA, and data applications, much finer bounds that take into account the vector space structure in a more refined way were needed. To do so, it helped to identify more refined structural properties that decoupled the random matrix ideas from the underlying linear algebraic ideas — understanding this will be central to the next two sections.

[4] By *heavy-tailed graph*, consider a graph — or equivalently the adjacency matrix of such a graph — in which quantities such as the degree distribution or eigenvalue distribution decay in a heavy-tailed or power law manner.

Although these newer algorithms identified more refined structural properties, they have the same general structure as the original randomized matrix algorithms. Recall that the general structure of the algorithms just reviewed is the following.

- Preprocess the input by: defining a non-uniformity structure over the columns of the input matrix; or performing a random projection/rotation to uniformize that structure.
- Draw a random sample of columns from the input matrix, either using the non-uniformity structure as an importance sampling distribution to select actual columns, or selecting columns uniformly at random in the rotated basis.
- Postprocess the sample with a traditional deterministic NLA method.

In the above algorithms, the preprocessing was very fast, in that the importance sampling distribution could be computed by simply passing over the data a few times from external storage; and the postprocessing consists of just computing the best rank-k or best rank-$2k$ approximation to the sample. As will become clear below, by making one or both of these steps more sophisticated, very substantially improved results can be obtained, both in theory and in practice. This can be accomplished, e.g., by using more sophisticated sampling probabilities or coupling the randomness in more sophisticated ways with traditional NLA methods, which in some cases will require additional computation.

4

Randomized Algorithms for Least-squares Approximation

In this section and the next, we will describe randomized matrix algorithms for the least-squares approximation and low-rank approximation problems. The analysis of low-rank matrix approximation algorithms described in Section 5 boils down to a randomized approximation algorithm for the least-squares approximation problem [30, 33, 76, 137]. For this reason, for pedagogical reasons, and due to the fundamental importance of the least-squares problem more generally, randomized algorithms for the least-squares problem will be the topic of this section.

4.1 Different Perspectives on Least-squares Approximation

Consider the problem of finding a vector x such that $Ax \approx b$, where the rows of A and elements of b correspond to constraints and the columns of A and elements of x correspond to variables. In the very *overconstrained least-squares approximation problem*, where the $m \times n$ matrix A has $m \gg n$,[1] there is in general no vector x such that $Ax = b$, and it is common to quantify "best" by looking for a vector x_{opt} such that the Euclidean norm of the residual error is small, i.e., to solve the

[1] In this section only, we will assume that $m \gg n$.

least-squares (LS) approximation problem

$$x_{opt} = \mathrm{argmin}_x ||Ax - b||_2. \tag{4.1}$$

This problem is ubiquitous in applications, where it often arises from fitting the parameters of a model to experimental data, and it is central to theory. Moreover, it has a natural statistical interpretation as providing the best estimator within a natural class of estimators; and it has a natural geometric interpretation as fitting the part of the vector b that resides in the column space of A. From the viewpoint of low-rank matrix approximation, this LS problem arises since measuring the error with a Frobenius or spectral norm, as in (2.2), amounts to choosing columns that are "good" in a least squares sense.[2]

There are a number of different perspectives one can adopt on this LS problem. Two major perspectives of central importance in this review are the following.

- **Algorithmic perspective.** From an algorithmic perspective, the relevant question is: how long does it take to compute x_{opt}? The answer to this question is that is takes $O(mn^2)$ time [98]. This can be accomplished with one of several algorithms — with the Cholesky decomposition (which is good if A has full column rank and is very well-conditioned); or with a variant of the QR decomposition (which is somewhat slower, but more numerically stable); or by computing the full SVD $A = U\Sigma V^T$ (which is often, but certainly not always, overkill, but which can be easier to explain[3]), and letting $x_{opt} = V\Sigma^+ U^T b$. Although these methods differ a great deal in practice and in terms of numerical

[2] Intuitively, these low-rank approximation algorithms find columns that provide a space that is good in a least-squares sense, when compared to the best rank-k space, at reconstructing every row of the input matrix. Thus, the reason for the connection is that the merit function that describes the quality of those algorithms is typically reconstruction error with respect to the spectral or Frobenius norm.

[3] The SVD has been described as the "Swiss Army Knife" of NLA [97]. That is, given it, one can do nearly anything one wants, but it is almost always overkill, as one rarely if ever needs its full functionality. Nevertheless, for pedagogical reasons, since other decompositions are typically better in terms of running time by only constant factors, and since numerically-stable algorithms for these latter decompositions can be quite complex, it is convenient to formulate results in terms of the SVD and the best rank-k approximation to the SVD.

implementation, asymptotically each of these methods takes a constant times mn^2 time to compute a vector x_{opt}. Thus, from an algorithmic perspective, a natural next question to ask is: can the general LS problem be solved, *either exactly or approximately*, in $o(mn^2)$ time,[4] with no assumptions at all on the input data?

- **Statistical perspective.** From a statistical perspective, the relevant question is: when is computing the x_{opt} the right thing to do? The answer to this question is that this LS optimization is the right problem to solve when the relationship between the "outcomes" and "predictors" is roughly linear and when the error processes generating the data are "nice" (in the sense that they have mean zero, constant variance, are uncorrelated, and are normally distributed; or when we have adequate sample size to rely on large sample theory) [50]. Thus, from a statistical perspective, a natural next question to ask is: what should one do when the assumptions underlying the use of LS methods are not satisfied or are only imperfectly satisfied?

Of course, there are also other perspectives that one can adopt. For example, from a numerical perspective, whether the algorithm is numerically stable, issues of forward versus backward stability, condition number issues, and whether the algorithm takes time that is a large or small constant multiplied by $\min\{mn^2, m^2n\}$ are of paramount importance.

When adopting the statistical perspective, it is common to check the extent to which the assumptions underlying the use of LS have been satisfied. To do so, it is common to assume that $b = Ax + \varepsilon$, where b is the response, the columns $A^{(i)}$ are the carriers, and ε is a "nice" error process.[5] Then $x_{opt} = (A^TA)^{-1}A^Tb$, and thus $\hat{b} = Hb$, where the

[4] Formally, $f(n) = o(g(n))$ as $n \to \infty$ means that for every positive constant ε there exists a constant N such that $|f(n)| \leq \varepsilon|g(n)|$, for all $n \geq N$. Informally, it means that $f(n)$ grows more slowly than $g(n)$. Thus, if the running time of an algorithm is $o(mn^2)$ time, then it is asymptotically faster than any (arbitrarily small) constant times mn^2.

[5] This is typically done by assuming that the error process ε consists of i.i.d. Gaussian entries. As with the construction of random projections in Section 3.1, numerous other

projection matrix onto the column space of A,

$$H = A(A^T A)^{-1} A^T,$$

is the so-called *hat matrix*. It is known that H_{ij} measures the influence or statistical leverage exerted on the prediction $\hat{b}_i$ by the observation b_j [110, 50, 49, 189, 51]. Relatedly, if the i^{th} diagonal element of H is particularly large then the i^{th} data point is particularly sensitive or influential in determining the best LS fit, thus justifying the interpretation of the elements H_{ii} as *statistical leverage scores* [137]. These leverage scores have been used extensively in classical regression diagnostics to identify potential outliers by, e.g., flagging data points with leverage score greater than 2 or 3 times the average value in order to be investigated as errors or potential outliers [50]. Moreover, in the context of recent graph theory applications, this concept has proven useful under the name of graph *resistance* [178]; and, for the matrix problems considered here, some researchers have used the term *coherence* to measure the degree of non-uniformity of these statistical leverage scores [41, 183, 132].

In order to compute these quantities *exactly*, recall that if U is *any* orthogonal matrix spanning the column space of A, then $H = P_U = UU^T$ and thus

$$H_{ii} = ||U_{(i)}||_2^2,$$

i.e., the statistical leverage scores equal the Euclidean norm of the *rows* of any such matrix U [76, 137]. Recall Definition 2.1 from Section 2.1. (Clearly, the columns of such a matrix U are orthonormal, but the rows of U in general are not — they can be uniform if, e.g., U consists of columns from a truncated Hadamard matrix; or extremely nonuniform if, e.g., the columns of U come from a truncated identity matrix; or anything in between.) More generally, and of interest for the low-rank matrix approximation algorithms in Section 5, the *statistical leverage*

constructions are possible and will yield similar results. Basically, it is important that no one or small number of data points has a particularly large influence on the LS fit, in order to ensure that techniques from large-sample theory like measure concentration apply.

scores relative to the best rank-k approximation to A are the diagonal elements of the projection matrix onto the best rank-k approximation to A. Thus, they can be computed from

$$(P_{U_k})_{ii} = ||U_{k,(i)}||_2^2,$$

where $U_{k,(i)}$ is the i^{th} row of any matrix spanning the space spanned by the top k left singular vectors of A (and similarly for the right singular subspace if columns rather than rows are of interest).

In many diagnostic applications, e.g., when one is interested in exploring and understanding the data to determine what would be the appropriate computations to perform, the time to compute or approximate $(P_U)_{ii}$ or $(P_{U_k})_{ii}$ is not the bottleneck. On the other hand, in cases where this time is the bottleneck, an algorithm we will describe in Section 4.4.2 will provide very fine approximations to all these leverage scores in time that is qualitatively faster than that required to compute them exactly.

4.2 A Simple Algorithm for Approximating Least-squares Approximation

Returning to the algorithmic perspective, consider the following random sampling algorithm for the LS approximation problem [75, 76]. Given a very overconstrained LS problem, where the input matrix A and vector b are *arbitrary*, but $m \gg n$:

- Compute the normalized statistical leverage scores $\{p_i\}_{i=1}^m$, i.e., compute $p_i = ||U_{(i)}||_2^2/n$, where U is the $m \times n$ matrix consisting of the left singular vectors of A.[6]

[6] Stating this in terms of the singular vectors is a convenience, but it can create confusion. In particular, although computing the SVD is sufficient, it is by no means necessary — here, U can be *any* orthogonal matrix spanning the column space of A [137]. Moreover, these probabilities are robust, in that any probabilities that are close to the leverage scores will suffice; see [69] for a discussion of approximately-optimal sampling probabilities. Finally, as we will describe below, these probabilities can be approximated quickly, i.e., more rapidly than the time needed to compute a basis exactly, or the matrix can be preprocessed quickly to make them nearly uniform.

- Randomly sample and rescale[7] $r = O(n \log n/\epsilon^2)$ constraints, i.e., rows of A and the corresponding elements of b, using these scores as an importance sampling distribution.
- Solve the induced subproblem $\tilde{x}_{opt} = \text{argmin}_x ||SAx - Sb||_2$, where the $r \times m$ matrix S represents the sampling-and-rescaling operation.

The induced subproblem can be solved using any appropriate direct or iterative LS solver as a black box. For example, one could compute the solution directly via the generalized inverse as $\tilde{x}_{opt} = (SA)^\dagger Sb$, which would take $O(rn^2)$ time [98]. Alternatively, one could use iterative methods such as the Conjugate Gradient Normal Residual method, which can produce an ϵ-approximation to the optimal solution of the sampled problem in $O(\kappa(SA) rn \log(1/\epsilon))$ time, where $\kappa(SA)$ is the condition number of SA [98]. As stated, this algorithm will compute all the statistical leverage scores exactly, and thus it will not be faster than the traditional algorithm for the LS problem — importantly, we will see below how to get around this problem.

Since this overconstrained[8] LS algorithm samples constraints and not variables, the dimensionality of the vector $\tilde{x}_{opt}$ that solves the subproblem is the same as that of the vector x_{opt} that solves the original problem. The algorithm just presented is described in more detail in [75, 76, 77], where it is shown that relative-error bounds of the form

$$||b - A\tilde{x}_{opt}||_2 \leq (1+\epsilon)||b - Ax_{opt}||_2 \quad \text{and} \tag{4.2}$$

$$||x_{opt} - \tilde{x}_{opt}||_2 \leq \sqrt{\epsilon}\left(\kappa(A)\sqrt{\gamma^{-2} - 1}\right)||x_{opt}||_2 \tag{4.3}$$

hold, where $\kappa(A)$ is the condition number of A and where $\gamma = ||UU^T b||_2/||b||_2$ is a parameter defining the amount of the mass of b

[7] Recall from the discussion in Section 3.1 that each sampled row should be rescaled by a factor of $1/rp_i$. Thus, it is these sampled-and-rescaled rows that enter into the subproblem that this algorithm constructs and solves.

[8] Not surprisingly, similar ideas apply to underconstrained LS problems, where $m \ll n$, and where the goal is to compute the minimum-length solution. In this case, one randomly samples columns, and one uses a somewhat more complicated procedure to construct an approximate solution, for which relative-error bounds also hold. In fact, as we will describe in Section 4.4.2, the quantities $\{p_i\}_{i=1}^m$ can be approximated in $o(mn^2)$ time by relating them to an underconstrained LS approximation problem and running such a procedure.

inside the column space of A.[9] Of course, there is randomization inside this algorithm, and it is possible to flip a fair coin "heads" 100 times in a row. Thus, as stated, with $r = O(n \log n/\epsilon^2)$, the randomized least-squares algorithm just described might fail with a probability δ that is no greater than a constant (say 1/2 or 1/10 or 1/100, depending on the (modest) constant hidden in the $O(\cdot)$ notation) that is independent of m and n. Of course, using standard methods [150], this failure probability can easily be improved to be an arbitrarily small δ failure probability. For example, this holds if $r = O(n \log(n) \log(1/\delta)/\epsilon^2)$ in the above algorithm; alternatively, it holds if one repeats the above algorithm $O(\log(1/\delta))$ times and keeps the best of the results.

As an aside, it is one thing for TCS researchers, well-known to be cavalier with respect to constants and even polynomial factors, to make such observations about big-O notation and the failure probability, but by now these facts are acknowledged more generally. For example, a recent review of coupling randomized low-rank matrix approximation algorithms with traditional NLA methods [107] starts with the following observation. "Our experience suggests that many practitioners of scientific computing view randomized algorithms as a desperate and final resort. Let us address this concern immediately. Classical Monte Carlo methods are highly sensitive to the random number generator and typically produce output with low and uncertain accuracy. In contrast, the algorithms discussed herein are relatively insensitive to the quality of randomness and produce highly accurate results. The probability of failure is a user-specified parameter that can be rendered negligible (say, less than 10^{-15}) with a nominal impact on the computational resources required."

Finally, it should be emphasized that modulo this failure probability δ that can be made arbitrarily small without adverse effect and an error ϵ that can also be made arbitrarily small, the above algorithm (as well as the basic low-rank matrix approximation algorithms of Section 5 that boil down to this randomized approximate LS algorithm) returns

[9] We should reemphasize that such relative-error bounds, either on the optimum value of the objective function, as in (4.2) and as is more typical in TCS, or on the vector or "certificate" achieving that optimum, as in (4.3) and as is of greater interest in NLA, provide an *extremely* strong notion of approximation.

an answer $\tilde{x}_{opt}$ that satisfies bounds of the form (4.2) and (4.3), independent of *any* assumptions at all on the input matrices A and b.

4.3 A Basic Structural Result

What the above random sampling algorithm highlights is that the "relevant non-uniformity structure" that needs to be dealt with in order to solve the LS approximation problem is defined by the statistical leverage scores. (Moreover, since the randomized algorithms for low-rank matrix approximation to be described in Section 5 boil down to a least-squares problem, an analogous statement is true for these low-rank approximation algorithms.) To see "why" this works, it is helpful to identify a deterministic structural condition sufficient for relative-error approximation — doing so decouples the linear algebraic part of the analysis from the randomized matrix part. This condition, implicit in the analysis of [75, 76], was made explicit in [77].

Consider preconditioning or premultiplying the input matrix A and the target vector b with some *arbitrary* matrix Z, and consider the solution to the LS approximation problem

$$\tilde{x}_{opt} = \text{argmin}_x ||Z(Ax-b)||_2. \tag{4.4}$$

Thus, for the random sampling algorithm described in Section 4.2, the matrix Z is a carefully-constructed data-dependent random sampling matrix, and for the random projection algorithm below it is a data-independent random projection, but more generally it could be *any* arbitrary matrix Z. Recall that the SVD of A is $A = U_A \Sigma_A V_A^T$; and, for notational simplicity, let $b^\perp = U_A^\perp {U_A^\perp}^T b$ denote the part of the right hand side vector b lying outside of the column space of A. Then, the following structural condition holds.

- **Structural condition underlying the randomized least-squares algorithm.** Under the assumption that Z satisfies the following two conditions:

$$\sigma_{min}^2(ZU_A) \geq 1/\sqrt{2}; \quad \text{and} \tag{4.5}$$

$$||U_A^T Z^T Z b^\perp||_2^2 \leq \frac{\epsilon}{2}||Ax_{opt} - b||_2^2, \tag{4.6}$$

for some $\epsilon \in (0,1)$, the solution vector $\tilde{x}_{opt}$ to the LS approximation problem (4.4) satisfies relative-error bounds of the form (4.2) and (4.3).

In this condition, the particular constants $1/\sqrt{2}$ and $1/2$ clearly don't matter — they have been chosen for ease of comparison with [77]. Also, recall that $||b^{\perp}||_2 = ||Ax_{opt} - b||_2$. Several things should be noted about these two structural conditions:

- First, since $\sigma_i(U_A) = 1$, for all $i \in [n]$, Condition (4.5) indicates that the rank of ZU_A is the same as that of U_A. Note that although Condition (4.5) only states that $\sigma_i^2(ZU_A) \geq 1/\sqrt{2}$, for all $i \in [n]$, for the randomized algorithm of Section 4.2, it will follow that $|1 - \sigma_i^2(ZU_A)| \leq 1 - 2^{-1/2}$, for all $i \in [n]$. Thus, one should think of Condition (4.5) as stating that ZU_A is an approximate isometry. Thus, this expression can be bounded with the approximate matrix multiplication spectral norm bound of (3.4).
- Second, since before preprocessing by Z, $b^{\perp} = U_A^{\perp} {U_A^{\perp}}^T b$ is clearly orthogonal to U_A, Condition (4.6) simply states that after preprocessing $Zb^{\perp}$ remains approximately orthogonal to ZU_A. Although Condition (4.6) depends on the right hand side vector b, the randomized algorithm of Section 4.2 satisfies it without using any information from b. The reason for not needing information from b is that the left hand side of Condition (4.6) is of the form of an approximate product of two different matrices — where for the randomized algorithm of Section 4.2 the importance sampling probabilities depend only on one of the two matrices — and thus one can apply an approximate matrix multiplication bound of the form (3.2).
- Third, as the previous two points indicate, Condition (4.5) and (4.6) both boil down to the problem of approximating the product of two matrices, and thus the algorithmic primitives on approximate matrix multiplication from Section 3.1 will be useful, either explicitly or within the analysis.

It should be emphasized that there is no randomization in these two structural conditions — they are deterministic statements about an arbitrary matrix Z that represent a structural condition sufficient for relative-error approximation. Of course, if Z happens to be a random matrix, e.g., representing a random projection or a random sampling process, then Conditions (4.5) or (4.6) may fail to be satisfied — but, conditioned on their being satisfied, the relative-error bounds of the form (4.2) and (4.3) follow. Thus, the effect of randomization enters only via Z, and it is decoupled from the linear algebraic structure.

4.4 Making this Algorithm Fast — in Theory

Given this structural insight, what one does with it depends on the application. In some cases, as in certain genetics applications [137, 161, 164] or when solving the Column Subset Selection Problem as described in Section 5.2, the time for the computation of the leverage scores is not the bottleneck, and thus they may be computed with the traditional procedure. In other cases, as when dealing with not-extremely-sparse random matrices or other situations where it is expected or hoped that no single data point is particularly influential, it is assumed that the scores are exactly or approximately uniform, in which case uniform sampling is appropriate. In general, however, the simple random sampling algorithm of Section 4.2 requires the computation of the normalized statistical leverage scores, and thus it runs in $O(mn^2)$ time.

In this subsection, we will describe *two* ways to speed up the random sampling algorithm of Section 4.2 so that it runs in $o(mn^2)$ time for arbitrary input. The first way involves preprocessing the input with a randomized Hadamard transform and then calling the algorithm of Section 4.2 with uniform sampling probabilities. The second way involves computing a quick approximation to the statistical leverage scores and then using those approximations as the importance sampling probabilities in the algorithm of Section 4.2. Both of these methods provide fast and accurate algorithms in theory, and straightforward extensions of them provide very fast and very accurate algorithms in practice.

4.4.1 A Fast Random Projection Algorithm for the LS Problem

Consider the following structured random projection algorithm for approximating the solution to the LS approximation problem.

- Premultiply A and b with an $n \times n$ randomized Hadamard transform HD.
- Uniformly sample roughly $r = O\left(n(\log n)(\log m) + \frac{n(\log m)}{\epsilon}\right)$ rows from HDA and the corresponding elements from HDb.
- Solve the induced subproblem $\tilde{x}_{opt} = \mathrm{argmin}_x ||SHDAx - SHDb||_2$, where the $r \times m$ matrix S represents the sampling operation.

This algorithm, which first preprocesses the input with a structured random projection and then solves the induced subproblem, as well as a variant of it that uses the original "fast" Johnson-Lindenstrauss transform [3, 4, 142], was presented in [77, 174], (where a precise statement of r is given and) where it is shown that relative-error bounds of the form (4.2) and (4.3) hold.

To understand this result, recall that premultiplication by a randomized Hadamard transform is a unitary operation and thus does not change the solution; and that from the SVD of A and of HDA it follows that $U_{HDA} = HDU_A$. Thus, the "right" importance sampling distribution for the preprocessed problem is defined by the diagonal elements of the projection matrix onto the span of HDU_A. Importantly, application of such a Hadamard transform tends to "uniformize" the leverage scores,[10] in the sense that all the leverage scores associated with U_{HDA} are (up to logarithmic fluctuations) uniform [3, 77]. Thus, uniform sampling probabilities are optimal, up to a logarithmic factor which can be absorbed into the sampling complexity. Overall, this relative-error approximation algorithm for the LS problem run in $o(mn^2)$ time [77, 174] — essentially $O\big(mn\log(n/\epsilon) + \frac{n^3 \log^2 m}{\epsilon}\big)$ time, which is much less than $O(mn^2)$ when $m \gg n$. Although the ideas

[10] As noted above, this is for very much the same reason that a Fourier matrix delocalizes a localized δ-function; and it also holds for the application of an unstructured random orthogonal matrix or random projection.

underlying the Fast Fourier Transform have been around and used in many applications for a long time [59, 103], they were first applied in the context of randomized matrix algorithms only recently [3, 77, 174].

The $o(mn^2)$ running time is most interesting when the input matrix to the overconstrained LS problem is dense; if the input matrix is sparse, then a more appropriate comparison might have to do with the number of nonzero entries. In general, however, random Gaussian projections and randomized Hadamard-based transforms tend not to respect sparsity. In some applications, e.g., the algorithm of [144] that is described in Section 4.5 and that automatically speeds up on sparse matrices and with fast linear operators, as well as several of the randomized algorithms for low-rank approximation to be described in Section 5.3, this can be worked around. In general, however, the question of using sparse random projections and sparsity-preserving random projections is an active topic of research [60, 94, 118, 119].

4.4.2 A Fast Random Sampling Algorithm for the LS Problem

Next, consider the following algorithm which takes as input an arbitrary $m \times n$ matrix A, with $m \gg n$, and which returns as output approximations to all m of the statistical leverage scores of A.

- Premultiply A by a structured random projection, e.g., $\Omega_1 = SHD$ from Section 3.1, which represents uniformly sampling roughly $r_1 = O(m \log n/\epsilon)$ rows from a randomized Hadamard transform.
- Compute the $m \times r_2$ matrix $X = A(\Omega_1 A)^\dagger \Omega_2$, where Ω_2 is an $r_1 \times r_2$ unstructured random projection matrix and where the dagger represents the Moore-Penrose pseudoinverse.
- For each $i = 1,\ldots,m$, compute and return $\tilde{\ell}_i = ||\tilde{X}_{(i)}||_2^2$.

This algorithm was introduced in [73], based on ideas in [134]. In [73], it is proven that

$$|\ell_i - \tilde{\ell}_i| \leq \epsilon \ell_i \quad \text{for all } i = 1,\ldots,m,$$

where ℓ_i are the statistical leverage scores of Definition 2.1. That is, this algorithm returns a relative-error approximation to *every* one of the m statistical leverage scores. Moreover, in [73] it is also proven that this algorithm runs in $o(mn^2)$ time — due to the structured random projection in the first step, the running time is basically the same time as that of the fast random projection algorithm described in the previous subsection.[11] In addition, given an arbitrary rank parameter k and an arbitrary-sized $m \times n$ matrix A, this algorithm can be extended to approximate the leverage scores relative to the best rank-k approximation to A in roughly $O(mnk)$ time. See [73] for a discussion of the technical issues associated with this. In particular, note that the problem of asking for approximations to the leverage scores relative to the best rank-k space of a matrix is ill-posed; and thus one must replace it by computing approximations to the leverage scores for some space that is a good approximation to the best rank-k space.

Within the analysis, this algorithm for computing rapid approximations to the statistical leverage scores of an arbitrary matrix basically boils down to an *under*constrained LS problem, in which a structured random projection is carefully applied, in a manner somewhat analogous to the fast *over*constrained LS random projection algorithm in the previous subsection. In particular, let A be an $m \times n$ matrix, with $m \ll n$, and consider the problem of finding the minimum-length solution to $x_{opt} = \text{argmin}_x ||Ax - b||_2 = A^+b$. Sampling variables or columns from A can be represented by postmultiplying A by a $n \times c$ (with $c > m$) column-sampling matrix S to construct the (still underconstrained) least-squares problem: $\tilde{x}_{opt} = \text{argmin}_x ||ASS^Tx - b||_2 = A^T(AS)^{T+}(AS)^+b$. The second equality follows by inserting $P_{A^T} = A^TA^{T+}$ to obtain $ASS^TA^TA^{T+}x - b$ inside the $|| \cdot ||_2$ and recalling that $A^+ = A^TA^{T+}A^+$ for the Moore-Penrose pseudoinverse. If one constructs S by randomly sampling $c = O((n/\epsilon^2)\log(n/\epsilon))$ columns according to "column-leverage-score" probabilities, i.e., exact or approximate diagonal elements of the projection matrix onto the row space of A, then it can be proven that $||x_{opt} - \tilde{x}_{opt}||_2 \leq \epsilon||x_{opt}||_2$ holds, with high probability. Alternatively,

[11] Recall that since the coherence of a matrix is equal to the largest leverage score, this algorithm also computes a relative-error approximation to the coherence of the matrix in $o(mn^2)$ time — which is qualitatively faster than the time needed to compute an orthogonal basis spanning the original matrix.

this underconstrained LS problem problem can also be solved with a random projection. By using ideas from the previous subsection, one can show that if S instead represents a random projection matrix, then by projecting to a low-dimensional space (which takes $o(m^2n)$ time with a structured random projection matrix), then relative-error approximation guarantees also hold.

Thus, one can run the following algorithm for approximating the solution to the general overconstrained LS approximation problem.

- Run the algorithm of this subsection to obtain numbers $\tilde{\ell}_i$, for each $i = 1,\ldots,m$, rescaling them to form a probability distribution over $\{1,\ldots,m\}$.
- Call the randomized LS algorithm that is described in Section 4.2, except using these numbers $\tilde{\ell}_i$ (rather than the exact statistical leverage scores ℓ_i) to construct the importance sampling distribution over the rows of A.

That is: approximate the normalized statistical leverage scores in $o(mn^2)$ time; and then call the random sampling algorithm of Section 4.2 using those approximate scores as the importance sampling distribution. Clearly, this combined algorithm provides relative-error guarantees of the form (4.2) and (4.3), and overall it runs in $o(mn^2)$ time.

4.4.3 Some Additional Thoughts

The previous two subsections have shown that the random sampling algorithm of Section 4.2, which naïvely needs $O(mn^2)$ to compute the importance sampling distribution, can be sped up in two different ways. One can spend $o(mn^2)$ to uniformize the sampling probabilities and then sample uniformly; or one can spend $o(mn^2)$ time to compute approximately the sampling probabilities and then use those approximations as an importance sampling distribution. Both procedures take basically the same time, which should not be surprising since the approximation of the statistical leverage scores boils down to an underconstrained LS problem; and which procedure is preferable in any given situation depends on the downstream application.

Finally, to understand better the relationship between random sampling and random projection algorithms for the least squares problem, consider the following vanilla random projection algorithm. Given a matrix A and vector b, representing a very overconstrained LS problem, one can:

- Construct an $O(n \log n/\epsilon^2) \times m$ random projection matrix Ω, where the entries consist (up to scaling) of i.i.d. Gaussians or $\{-1, +1\}$.
- Solve the induced subproblem $\tilde{x}_{opt} = \text{argmin}_x ||\Omega Ax - \Omega b||_2$.

It is relatively-easy to show that this algorithm also satisfies relative-error bounds of the form (4.2) and (4.3). Importantly, though, this random projection algorithm requires $O(mn^2)$ time to implement. For the random sampling algorithm of Section 4.2, the $O(mn^2)$ computational bottleneck is computing the importance sampling probabilities; while for this random projection algorithm the $O(mn^2)$ computational bottleneck is actually performing the matrix-matrix multiplication representing the random projection. Thus, one can view the $o(mn^2)$ random projection algorithm that uses a small number of rows from a randomized Hadamard transform in one of two ways: either as a structured approximation to a usual random projection matrix with i.i.d. Gaussian or $\{-1, +1\}$ entries that can be rapidly applied to arbitrary vectors; or as preprocessing the input with an orthogonal transformation to uniformize the leverage scores so that uniform sampling is appropriate.

4.5 Making this Algorithm Fast — in Practice

Several "rubber-hits-the-road" issues need to be dealt with in order for the algorithms of the previous subsection to yield to high-precision numerical implementations that beat traditional numerical code, either in specialized scientific applications or when compared with popular numerical libraries. The following issues are most significant.

- **Awkward ϵ dependence.** The sampling complexity, i.e., the number of columns or rows needed by the algorithm, scales as $1/\epsilon^2$ or $1/\epsilon$, which is the usual poor asymptotic complexity for Monte Carlo methods. Even though there exist lower bounds for these problems for certain models of data access [55], this is problematic if one wants to choose ϵ to be on the order of machine precision.

- **Numerical conditioning and preconditioning.** Thus far, nothing has been said about numerical conditioning issues, although it is well-known that these issues are crucial when matrix algorithms are implemented numerically.
- **Forward error versus backward error.** The bounds above, e.g., (4.2) and (4.3), provide so-called forward error bounds. The standard stability analysis in NLA is in terms of the backward error, where the approximate solution $\tilde{x}_{opt}$ is shown to be the exact solution of some slightly perturbed problem $x_{opt} = \text{argmin}_x ||(A + \delta A)x - b||_2$, where $||\delta A|| \leq \tilde{\epsilon}||A||$ for some small $\tilde{\epsilon}$.

All three of these issues can be dealt with by coupling the randomized algorithms of the previous subsection with traditional tools from iterative NLA algorithms (as opposed to applying a traditional NLA algorithm as a black box on the random sample or random projection, as the above algorithms do). This was first done by [169], and these issues were addressed in much greater detail by [9] and then by [56] and [144].

Both of the implementations of [169, 9] take the following form.

- Premultiply A by a structured random projection, e.g., $\Omega = SHD$ from Section 3.1, which represents uniformly sampling a few rows from a randomized Hadamard transform.
- Perform a QR decomposition on ΩA, so that $\Omega A = QR$.
- Use the R from the QR decomposition as a preconditioner for an iterative Krylov-subspace [98] method.

In general, iterative algorithms compute an ϵ-approximate solution to a LS problem like (4.1) by performing $O(\kappa(A)\log(1/\epsilon))$ iterations, where $\kappa(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$ is the condition number of the input matrix (which could be quite large, thereby leading to slow convergence of the iterative algorithm).[12] In this case, by choosing the dimension of

[12] These iterative algorithms replace the solution of (4.1) with two problems: first solve $x_{opt} = \text{argmin}_x ||A\Pi^{-1}y - b||_2$ iteratively, where Π is the preconditioner; and then solve

the random projection appropriately, e.g., as discussed in the previous subsections, one can show that $\kappa(AR^{-1})$ is bounded above by a small data-independent constant. That is, by using the R matrix from a QR decomposition of ΩA, one obtains a good preconditioner for the original problem (4.1), independent of course of any assumptions on the original matrix A. Overall, applying the structured random projection in the first step takes $o(mn^2)$ time; performing a QR decomposition of ΩA is fast since ΩA is much smaller than A; and one needs to perform only $O(\log(1/\epsilon))$ iterations, each of which needs $O(mn)$ time, to compute the approximation.

The algorithm of [169] used CGLS (Conjugate Gradient Least Squares) as the iterative Krylov-subspace method, while the algorithm of [9] used LSQR [155]; and both demonstrate that randomized algorithms can outperform traditional deterministic NLA algorithms in terms of clock-time (for particular implementations or compared with Lapack) for computing high-precision solutions for LS systems with as few as thousands of constraints and hundreds of variables. The algorithm of [9] considered five different classes of structured random projections (i.e., Discrete Fourier Transform, Discrete Cosine Transform, Discrete Hartely Transform, Walsh-Hadamard Transform, and a Kac random walk), explicitly addressed conditioning and backward stability issues, and compared their implementation with Lapack on a wide range of matrices with uniform, moderately nonuniform, and very nonuniform leverage score importance sampling distributions. Similar ideas have also been applied to other common NLA tasks; for example, [56] shows that these ideas can be used to develop fast randomized algorithms for computing projections onto the null space and row space of A, for structured matrices A such that both A and A^T can be rapidly applied to arbitrary vectors.

The implementations of [56, 144] are similar, except that the random projection matrix in the first step of the above procedure is a traditional Gaussian random projection matrix. While this does *not* lead to a $o(mn^2)$ running time, it can be appropriate in certain situations: for

$\Pi x = y$. Thus, a matrix Π is a good preconditioner if $\kappa(A\Pi^{-1})$ is small and if $\Pi x = y$ can be solved quickly.

example, if both A and its adjoint A^T are structured such that they can be applied rapidly to arbitrary vectors [56]; or for solving large-scale problems on distributed clusters with high communication cost [144]. For example, due to the Gaussian random projection, the preconditioning phase of the algorithm of [144] is very well-conditioned, which implies that the number of iterations is fully predictable when LSQR or the Chebyshev semi-iterative method is applied to the preconditioned system. The latter method is more appropriate for parallel computing environments, where communication is a major issue, and thus [144] illustrates the empirical behavior of the algorithm on Amazon Elastic Compute Cloud (EC2) clusters.

5

Randomized Algorithms for Low-rank Matrix Approximation

In this section, we will describe several related randomized algorithms for low-rank matrix approximation that underlie applications such as those described in Section 2. These algorithms build on the algorithms of Section 3.2; and they achieve much-improved worst-case bounds and are more useful in both numerical analysis and data analysis applications (when compared with the algorithms of Section 3.2). First, the algorithm of Section 5.1 is a random sampling algorithm that improves the additive-error bounds to much finer relative-error bounds — the analysis of this algorithm boils down to an immediate application of the least-squares approximation algorithm of Section 4. Next, the algorithm of Section 5.2 is a random sampling algorithm that returns *exactly* k, rather than $O(k \log k/\epsilon^2)$, columns, for an input rank parameter k — the proof of this result involves a structural property that decouples the randomization from the linear algebra in a somewhat more refined way than we saw in Section 4. Finally, the algorithms of Section 5.3 are random projection algorithms that take advantage of this more refined structural property to couple these randomized algorithms with traditional methods from NLA and scientific computing.[1]

[1] Most of the results in this section will be formulated in terms of the amount of spectral or Frobenius norm that is captured by the (full or rank-k approximation to the) random

5.1 A Basic Random Sampling Algorithm

Additive-error bounds (of the form proved for the low-rank algorithms of Section 3.2) are rather coarse, and the gold standard in TCS is to establish much finer relative-error bounds of the form provided in (5.1) below. To motivate the importance sampling probabilities used to achieve such relative-error guarantees, recall that if one is considering a matrix with $k-1$ large singular values and one much smaller singular value, then the directional information of the k^{th} singular direction will be hidden from the Euclidean norms of the input matrix. The reason is that, since $A_k = U_k \Sigma_k V_k^T$, the Euclidean norms of the columns of A are convolutions of "subspace information" (encoded in U_k and V_k^T) and "size-of-A information" (encoded in Σ_k). This *suggests* deconvoluting subspace information and size-of-A information by choosing importance sampling probabilities that depend on the Euclidean norms of the columns of V_k^T. This importance sampling distribution defines a non-uniformity structure over $\mathbb{R}^n$ that indicates *where* in the n-dimensional space the information in A is being sent, independent of *what* that (singular value) information is. More formally, these quantities are proportional to the diagonal elements of the projection matrix onto the span of V_k^T,[2] and thus they are examples of generalized statistical leverage scores.

This idea was suggested in [76, 137], and it forms the basis for the algorithm from the TCS literature that achieves the strongest Frobenius norm bounds.[3] Given an $m \times n$ matrix A and a rank

sample or random projection. Given a basis for this sample or projection, it is straightforward to compute other common decompositions such as the pivoted QR factorization, the eigenvalue decomposition, the partial SVD, etc. using traditional NLA methods; see [107] for a good discussion of this.

[2] Here, we are sampling columns and not rows, as in the algorithms of Section 4, and thus we are dealing with the right, rather than the left, singular subspace; but clearly the ideas are analogous. Thus, in particular, note that the "span of V_k^T" refers to the span of the *rows* of V_k^T, whereas the "span of U_k," as used in previous sections, refers to the span of the *columns* of U_k.

[3] Subsequent work in TCS that has not (yet?) found application in NLA and data analysis also achieves similar relative-error bounds but with different methods. For example, the algorithm of [109] runs in roughly $O(mnk^2 \log k)$ time and uses geometric ideas involving sampling and merging approximately optimal k-flats. Similarly, the algorithm of [64] randomly samples in a more complicated manner and runs in $O(Mk^2 \log k)$, where M is the

parameter k:

- Compute the importance sampling probabilities $\{p_i\}_{i=1}^{n}$, where $p_i = \frac{1}{k}||V_k^{T^{(i)}}||_2^2$, where V_k^T is *any* $k \times n$ orthogonal matrix spanning the top-k right singular subspace of A.
- Randomly select and rescale $c = O(k \log k/\epsilon^2)$ columns of A according to these probabilities to form the matrix C.

A more detailed description of this basic random sampling algorithm may be found in [76, 137], where it is proven that

$$||A - P_{C_k}A||_F \leq (1+\epsilon)||A - P_{U_k}A||_F \tag{5.1}$$

holds. (As above, C_k is the best rank-k approximation to the matrix C, and P_{C_k} is the projection matrix onto this k-dimensional space.) As with the relative-error random sampling algorithm of Section 4.2, the dependence of the sampling complexity and running time on the failure probability δ is $O(\log(1/\delta))$; thus, the failure probability for this randomized low-rank approximation, as well as the subsequent algorithms of this section, can be made to be negligibly-small, both in theory and in practice. The analysis of this algorithm boils down to choosing a set of columns that are relative-error good at capturing the Frobenius norm of A, when compared to the basis provided by the top-k singular vectors. That is, it boils down to the randomized algorithm for the least-squares approximation problem from Section 4; see [76, 137] for details.

This algorithm and related algorithms that randomly sample columns and/or rows provide what is known as *CX or CUR matrix decompositions* [71, 76, 137].[4] In addition, this relative-error column-based CUR decomposition, as well as heuristic variants of it, has been

number of nonzero elements in the matrix; alternatively, it runs in $O(k \log k)$ passes over the data from external storage.

[4] Within the NLA community, Stewart developed the quasi-Gram-Schmidt method and applied it to a matrix and its transpose to obtain such a CUR matrix decomposition [179, 23]; and Goreinov, Tyrtyshnikov, and Zamarashkin developed a CUR matrix decomposition (a so-called pseudoskeleton approximation) and related the choice of columns and rows to a "maximum uncorrelatedness" concept [100, 99]. Note that the Nyström method is a type of CUR decomposition and that the pseudoskeleton approximation is also a generalization of the Nyström method.

applied in a range of data analysis applications, ranging from term-document data to DNA SNP data [137, 163, 164]. The computational bottleneck for this relative-error approximation algorithm is computing the importance sampling distribution $\{p_i\}_{i=1}^n$, for which it suffices to compute *any* $k \times n$ matrix V_k^T that spans the top-k right singular subspace of A. That is, it *suffices* (but is *not* necessary) to compute any orthonormal basis spanning V_k^T, which typically requires $O(mnk)$ running time, and it is *not* necessary to compute the full or partial SVD. Alternatively, the leverage scores can all be approximated to within $1 \pm \epsilon$ in roughly $O(mnk)$ time using the algorithm of [73] from Section 4.4.2, in which case these approximations can be used as importance sampling probabilities in the above random sampling algorithm.

5.2 A More Refined Random Sampling Algorithm

The algorithm of the previous subsection randomly samples $O(k \log k/\epsilon^2)$ columns, and then in order to compare with the bound provided by the SVD, it "filters" those columns through an exactly rank-k space. In this subsection, we will describe a randomized algorithm for the Column Subset Selection Problem (CSSP): the problem of selecting *exactly* k columns from an input matrix. Clearly, bounds for algorithms for the CSSP will be worse than the very strong relative-error bounds, provided by (5.1), that hold when $O(k \log k/\epsilon^2)$ columns are selected. Importantly, though, this CSSP algorithm extends the ideas of the previous relative-error TCS algorithm to obtain bounds of the form used historically in NLA. Moreover, the main structural result underlying the analysis of this algorithm permits *much* finer control on the application of randomization, e.g., for high-performance numerical implementation of both random sampling and random projection algorithms. We will start with a review of prior approaches to the CSSP; then describe the algorithm and the quality-of-approximation bounds; and finally highlight the main structural result underlying its analysis.

5.2.1 A Formalization of and Prior Approaches to this Problem

Within NLA, a great deal of work has focused on this CSSP [25, 27, 38, 44, 46, 47, 86, 105, 111, 156, 157]. Several general observations about the NLA approach include:

- The focus in NLA is on *deterministic algorithms.* Moreover, these algorithms are greedy, in that at each iterative step, the algorithm makes a decision about which columns to keep according to a pivot-rule that depends on the columns it currently has, the spectrum of those columns, etc. Differences between different algorithms often boil down to how to deal with such pivot rules decisions, and the hope is that more sophisticated pivot-rule decisions lead to better algorithms in theory or in practice.
- There are deep *connections with QR factorizations* and in particular with the so-called Rank Revealing QR factorizations. Moreover, there is an emphasis on optimal conditioning questions, backward error analysis issues, and whether the running time is a large or small constant multiplied by n^2 or n^3.
- Good *spectral norm bounds* are obtained. A typical spectral norm bound is:

  $$||A - P_C A||_2 \leq O(\sqrt{k(n-k)})||A - P_{U_k} A||_2, \tag{5.2}$$

  and these results are algorithmic, in that the running time is a low-degree polynomial in m and n [105]. (The $\sqrt{n}$ multiplicative factor might seem large, but recall that the spectral norm is much less “forgiving” than the Frobenius norm and that it is not even known whether there exists columns that do better.) On the other hand, the strongest result for the Frobenius norm in this literature is

  $$||A - P_C A||_F \leq \sqrt{(k+1)(n-k)}||A - P_{U_k} A||_2, \tag{5.3}$$

  but it is only an existential result, i.e., the only known algorithm essentially involves exhaustive enumeration [111]. (In these two expressions, U_k is the $m \times k$ matrix consisting of the top k left singular vectors of A, and P_{U_k} is a projection matrix onto the span of U_k.)

Within TCS, a great deal of work has focused on the related problem of choosing good columns from a matrix [68, 69, 70, 71, 76, 172]. Several general observations about the TCS approach include:

- The focus in TCS is on *randomized algorithms*. In particular, with these algorithms, there exists some nonzero probability, which can typically be made extremely small, say $\delta = 10^{-20}$, that the algorithm will return columns that fail to satisfy the desired quality-of-approximation bound.
- The algorithms select *more than k columns*, and the best rank-k projection onto those columns is considered. The number of columns is typically a low-degree polynomial in k, most often $O(k \log k)$, where the constants hidden in the big-O notation are quite reasonable.
- Very good *Frobenius norm bounds* are obtained. For example, the algorithm (described above) that provides the strongest Frobenius norm bound achieves (5.1), while running in time of the order of computing an exact or approximate basis for the top-k right singular subspace [76]. The TCS literature also demonstrates that there exists a set of k columns that achieves a constant-factor approximation:
$$||A - P_C A||_F \leq \sqrt{k}||A - P_{U_k} A||_F, \tag{5.4}$$
but note that this is an existential result [65].

Note that, prior to the algorithm of the next subsection, it was not immediately clear how to combine these two very different approaches. For example, if one looks at the details of the pivot rules in the deterministic NLA methods, it isn't clear that keeping more than exactly k columns will help at all in terms of reconstruction error. Similarly, since there is a version of the so-called Coupon Collector Problem at the heart of the usual TCS analysis, keeping fewer than $\Omega(k \log k)$ will fail with respect to this worst-case analysis. Moreover, the obvious hybrid algorithm of first randomly sampling $O(k \log k)$ columns and then using a deterministic QR procedure to select exactly k of those columns does not seem to perform so well (either in theory or in practice).

5.2.2 A Two-stage Hybrid Algorithm for this Problem

Consider the following more sophisticated version of a two-stage hybrid algorithm. Given an arbitrary $m \times n$ matrix A and rank parameter k:

- (Randomized phase) Compute the importance sampling probabilities $\{p_i\}_{i=1}^n$, where $p_i = \frac{1}{k}||V_k^{T^{(i)}}||_2^2$, where V_k^T is *any* $k \times n$ orthogonal matrix spanning the top-k right singular subspace of A. Randomly select and rescale $c = O(k \log k)$ columns of V_k^T according to these probabilities.
- (Deterministic phase) Let $\tilde{V}^T$ be the $k \times O(k \log k)$ non-orthogonal matrix consisting of the down-sampled and rescaled columns of V_k^T. Run a deterministic QR algorithm on $\tilde{V}^T$ to select exactly k columns of $\tilde{V}^T$. Return the corresponding columns of A.

A more detailed description of this algorithm may be found in [30, 33], where it is shown that with extremely high probability the following spectral[5] and Frobenius norm bounds hold:

$$||A - P_C A||_2 \leq O(k^{3/4} \log^{1/2}(k) n^{1/2}) ||A - P_{U_k} A||_2 \qquad (5.5)$$

$$||A - P_C A||_F \leq O(k \log^{1/2} k) ||A - P_{U_k} A||_F. \qquad (5.6)$$

Note that both the original choice of columns in the first phase, as well as the application of the QR algorithm in the second phase,[6] involve the sampled version of the matrix V_k^T, i.e., the matrix defining the relevant non-uniformity structure over the columns of A in the relative-error algorithm of Section 5.1. In particular, it is critical to the success of this algorithm that the QR procedure in the second phase be applied to the randomly-sampled version of V_k^T, rather than of A itself. This algorithm may be viewed as post-processing the relative-error random sampling algorithm of the previous subsection to remove redundant

[5] Note that to establish the spectral norm bound, [30, 33] used slightly more complicated (but still depending only on information in V_k^T) importance sampling probabilities, but this may be an artifact of the analysis.

[6] Note that QR (as opposed to the SVD) is *not* performed in the second phase to speed up the computation of a relatively cheap part of the algorithm, but instead it is performed since the goal of the algorithm is to return actual columns of the input matrix.

columns; and it has been applied successfully to a range of data analysis problems. See, e.g., [72, 161, 162] and [31, 32, 176], as well as [35] for a discussion of numerical issues associated with this algorithm.

With respect to running time, the computational bottleneck for this algorithm is computing $\{p_i\}_{i=1}^n$, for which it suffices to compute *any* $k \times n$ matrix V_k^T that spans the top-k right singular subspace of A. (In particular, a full or partial SVD computation is *not* necessary.) Thus, this running time is of the same order as the running time of the QR algorithm used in the second phase when applied to the original matrix A, typically roughly $O(mnk)$ time. (Not surprisingly, one could also perform a random projection, such as those described in Section 5.3 below, to approximate this basis, and then use that approximate basis to compute approximate importance sampling probabilities, as described in Section 4.4.2 above. In that case, similar bounds would hold, but the running time would be improved to $O(mn \log k)$ time.) Moreover, this algorithm scales up to matrices with thousands of rows and millions of columns, whereas existing off-the-shelf implementations of traditional QR algorithms may fail to run at all. With respect to the worst-case quality of approximation bounds, this algorithm selects columns that are comparable to the state-of-the-art algorithms for constant k (i.e., (5.5) is only $O(k^{1/4} \log^{1/2} k)$ worse than (5.2)) for the spectral norm; and (5.6) is only a factor of at most $O((k \log k)^{1/2})$ worse than (5.4), the best previously-known existential result for the Frobenius norm.

5.2.3 A Basic Structural Result

As with the relative-error LS algorithm of Section 4, in order to see "why" this algorithm for the CSSP works, it is helpful to identify a structural condition that decouples the randomization from the linear algebra. This structural condition was first identified in [30, 33], and it was subsequently improved by [107]. To identify it, consider preconditioning or postmultiplying the input matrix A by some *arbitrary* matrix Z. Thus, for the above randomized algorithm, the matrix Z is a carefully-constructed random sampling matrix, but it could be a random projection, or more generally *any* other arbitrary matrix Z.

Recall that if $k \leq r = \mathrm{rank}(A)$, then the SVD of A may be written as

$$A = U_A \Sigma_A V_A^T = U_k \Sigma_k V_k^T + U_{k,\perp} \Sigma_{k,\perp} V_{k,\perp}^T,$$

where U_k is the $m \times k$ matrix consisting of the top k singular vectors, $U_{k,\perp}$ is the $m \times (r-k)$ matrix consisting of the bottom $r-k$ singular vectors, etc. Then, the following structural condition holds.

- **Structural condition underlying the randomized low-rank algorithm.** If $V_k^T Z$ has full rank, then for $\nu \in \{2, F\}$, i.e., for both the Frobenius and spectral norms,

  $$||A - P_{AZ}A||_\nu^2 \leq ||A - A_k||_\nu^2 + \left|\left|\Sigma_{k,\perp} \left(V_{k,\perp}^T Z\right) \left(V_k^T Z\right)^\dagger \right|\right|_\nu^2 \tag{5.7}$$

  holds, where P_{AZ} is a projection onto the span of AZ, and where the dagger symbol represents the Moore-Penrose pseudoinverse.

This structural condition characterizes the manner in which the behavior of the low-rank algorithm depends on the interaction between the right singular vectors of the input matrix and the matrix Z. (In particular, it depends on the interaction between the subspace associated with the top part of the spectrum and the subspace associated with the bottom part of the spectrum via the $(V_{k,\perp}^T Z)(V_k^T Z)^\dagger$ term.) Note that the assumption that $V_k^T Z$ does not lose rank is a generalization of Condition (4.6) of Section 4. Also, note that the form of this structural condition is the same for both the spectral and Frobenius norms.

As with the LS problem, given this structural insight, what one does with it depends on the application: one can compute the basis V_k^T exactly if that is not computationally prohibitive and if one is interested in extracting exactly k columns; or one can perform a random projection and ensure that with high probability the structural condition is satisfied. Moreover, by decoupling the randomization from the linear algebra, it is easier to parameterize the problem in terms more familiar to NLA and scientific computing: for example, one can consider sampling $\ell > k$ columns and projecting onto a rank-k', where $k' > k$, approximation to those columns; or one can couple these ideas with traditional methods such as the power iteration method. Several of these extensions will be the topic of the next subsection.

5.3 Several Related Random Projection Algorithms

In this subsection, we will describe three random projection algorithms that draw on the ideas of the previous subsections in progressively finer ways.

5.3.1 A Basic Random Projection Algorithm

To start, consider the following basic random projection algorithm. Given an $m \times n$ matrix A and a rank parameter k:

- Construct an $n \times \ell$, with $\ell = O(k/\epsilon)$, structured random projection matrix Ω, e.g., $\Omega = DHS$ from Section 3.1, which represents uniformly sampling a few rows from a randomized Hadamard transform.
- Return $B = A\Omega$.

This algorithm, which amounts to choosing uniformly a small number ℓ of columns in a randomly rotated basis, was introduced in [174], where it is proven that

$$||A - P_{B_k}A||_F \leq (1+\epsilon)||A - P_{U_k}A||_F \tag{5.8}$$

holds with high probability. (Recall that B_k is the best rank-k approximation to the matrix B, and P_{B_k} is the projection matrix onto this k-dimensional space.) This algorithm runs in $O(Mk/\epsilon + (m+n)k^2/\epsilon^2)$ time, where M is the number of nonzero elements in A, and it requires 2 passes over the data from external storage.

Although this algorithm is very similar to the additive-error random projection algorithm of [159] that was described in Section 3.2, this algorithm achieves much stronger relative-error bounds by performing a much more refined analysis. Basically, [174] (and also the improvement [153]) modifies the analysis of the relative-error random sampling of [76, 137] that was described in Section 5.1, which in turn relies on the relative-error random sampling algorithm for LS approximation [75, 76] that was described in Section 4. In the same way that we saw in Section 4.4 that fast structured random projections could be used to uniformize coordinate-based non-uniformity structure for the LS problem, after which fast uniform sampling was appropriate, here uniform sampling in the randomly-rotated basis achieves relative-error bounds. In showing this, [174] also states a "subspace" analogue to the JL lemma, in which the geometry of an entire subspace of vectors (rather than just N pairs of vectors) is preserved. Thus, one can view the analysis of [174] as applying JL ideas, not to the rows of A itself, as was done by [159], but instead to vectors defining the subspace structure of A. Thus, with this random projection algorithm, the subspace information and size-of-A information are deconvoluted *within the analysis*, whereas with the random sampling algorithm of Section 5.1, this took place *within the algorithm* by modifying the importance sampling probabilities.

5.3.2 An Improved Random Projection Algorithm

As with the randomized algorithms for the LS problem, several rubber-hits-the-road issues need to be dealt with in order for randomized algorithms for the low-rank matrix approximation problem to yield to high-precision numerical implementations that beat traditional deterministic numerical code. In addition to the issues described in Section 4.5, the main issue here is the following.

- **Minimizing the oversampling factor.** In practice, choosing even $O(k \log k)$ columns, in either the original or a randomly-rotated basis, even when the big-O notation hides only modest constants, can make it difficult for these randomized matrix algorithms to beat previously-existing high-quality numerical implementations. Ideally, one could parameterize the problem so as to choose some number $\ell = k + p$ columns, where p is a modest additive oversampling factor, e.g., 10 or 20 or k, and where there is no big-O constant.

When attempting to be this aggressive at minimizing the size of the sample, the choice of the oversampling factor p is more sensitive to the input than in the algorithms we have reviewed so far. That is, whereas the previous bounds held for any input, here the proper choice for the oversampling factor p can be quite sensitive to the input matrix. For example, when parameterized this way, p could depend on the size of the matrix dimensions, the decay properties of the spectrum, and the particular choice made for the random projection matrix [106, 107, 128, 141, 168, 193]. Moreover, for worst-case input matrices, such a procedure may fail. For example, one can very-easily construct matrices such that if one randomly samples $o(k \log k)$ columns, in either the original canonical basis or in the randomly-rotated basis provided by the structured Hadamard transform, then the algorithm will fail. Basically, one can easily encode the so-called Coupon Collector Problem [150] in the columns, and it is known that $\Theta(k \log k)$ samples are necessary for this problem.

That being said, running the risk of such a failure might be acceptable if one can efficiently couple to a diagnostic to check for such a failure, and if one can then correct for it by choosing more samples if necessary. The best numerical implementations of randomized matrix algorithms for low-rank matrix approximation do just this, and the strongest results in terms of minimizing p take advantage of Condition (5.7) in a somewhat different way than was originally used in the analysis of the CSSP [107]. For example, rather than choosing $O(k \log k)$ dimensions and then filtering them through *exactly* k dimensions, as the relative-error random sampling and relative-error random projection algorithms do, one can choose some number ℓ of dimensions and project onto a k'-dimensional subspace, where $k < k' \leq \ell$, while exploiting Condition (5.7) to bound the error, as appropriate for the computational environment at hand [107].

Next, consider a second random projection algorithm that will address this issue. Given an $m \times n$ matrix A, a rank parameter k, and an oversampling factor p:

- Set $\ell = k + p$.
- Construct an $n \times \ell$ random projection matrix Ω, either with i.i.d. Gaussian entries or in the form of a structured

random projection such as $\Omega = DHS$ which represents uniformly sampling a few rows from a randomized Hadamard transform.
- Return $B = A\Omega$

Although this is quite similar to the algorithms of [159, 174], algorithms parameterized in this form were introduced in [128, 141, 193], where a suite of bounds of the form

$$||A - Z||_2 \lesssim 10\sqrt{\ell \min\{m,n\}}||A - A_k||_2$$

are shown to hold with high probability. Here, Z is a rank-k-or-greater matrix easily-constructed from B. This result can be used to obtain a so-called *interpolative decomposition* (a variant of the basic CSSP with explicit numerical conditioning properties), and [128, 141, 193] also provide an *a posteriori* error estimate (that is useful for situations in which one wants to choose the rank parameter k to be the numerical rank, as opposed to the *a priori* specification of k as part of the input, which is more common in the TCS-style algorithms that preceded this algorithm).

5.3.3 A Third Random Projection Algorithm

Finally, consider a third random projection algorithm that will address the issue that the decay properties of the spectrum can be important when it is of interest to minimize the oversampling very aggressively.[7] Given an $m \times n$ matrix A, a rank parameter k, an oversampling factor p, and an iteration parameter q:

- Set $\ell = k + p$.
- Construct an $n \times \ell$ random projection matrix Ω, either with i.i.d. Gaussian entries or in the form of a structured random projection such as $\Omega = DHS$ which represents

[7] Of course, this should not be completely unexpected, given that Condition (5.7) shows that the behavior of algorithms depends on the interaction between different subspaces associated with the input matrix A. When stronger assumptions are made about the data, stronger bounds can often be obtained.

uniformly sampling a few rows from a randomized Hadamard transform.

- Return $B = (AA^T)^q A\Omega$

This algorithm (as well as a numerically-stable variant of it) was introduced in [168], where it is shown that bounds of the form

$$||A - Z||_2 \lesssim \left(10\sqrt{\ell \min\{m,n\}}\right)^{1/(4q+2)} ||A - A_k||_2$$

hold with high probability. (This bound should be compared with the bound for the previous algorithm, and thus Z is a rank-k-or-greater matrix easily-constructed from B.) Basically, this random projection algorithm modifies the previous algorithm by coupling a form of the power iteration method within the random projection step. This has the effect of speeding up the decay of the spectrum while leaving the singular vectors unchanged, and it is observed in [107, 168] that $q = 2$ or $q = 4$ is often sufficient for certain data matrices of interest. This algorithm was analyzed in greater detail for the case of Gaussian random matrices in [107], and an out-of-core implementation (meaning, appropriate for data sets that are too large to be stored in RAM) of it was presented in [106].

The running time of these last two random projection algorithms depends on the details of the computational environment, e.g., whether the matrix is large and dense but fits into RAM or is large and sparse or is too large to fit into RAM; how precisely the random projection matrix is constructed; whether the random projection is being applied to an arbitrary matrix A or to structured input matrices, etc. [107]. For example, if random projection matrix Ω is constructed from i.i.d. Gaussian entries then in general the algorithm requires $O(mnk)$ time to implement the random projection, i.e., to perform the matrix-matrix multiplication $A\Omega$, which is no faster than traditional deterministic methods. On the other hand, if the projection is to be applied to matrices A such that A and/or A^T can be applied rapidly to arbitrary vectors (e.g., very sparse matrices, or structured matrices such as those arising from Toeplitz operators, or matrices that arise from discretized integral operators that can be applied via the fast multipole method), then Gaussian random projections may be preferable. Similarly, in general, if Ω is structured, e.g., is of the form $\Omega = DHS$, then it can be implemented in $O(mn \log k)$ time, and this can lead to dramatic clock-time speed-up over classical techniques even for problems of moderate sizes. On the other hand, for out-of-core implementations these additional speed-ups have a negligible effect, e.g., since matrix-matrix multiplications can be faster than a QR factorization, and so using Gaussian projections can be preferable. Working through these issues in theory and practice is still very much an active research area.

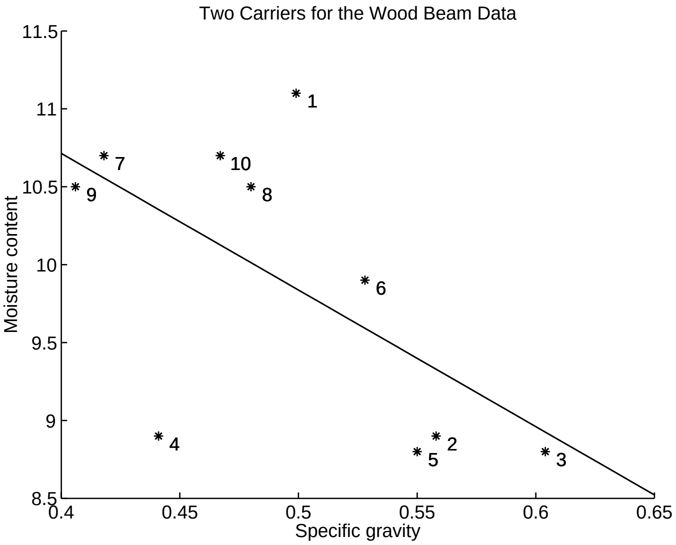
6

Empirical Observations

In this section, we will make some empirical observations, with an emphasis on the role of statistical leverage in these algorithms and in MMDS applications more generally.

6.1 Traditional Perspectives on Statistical Leverage

As mentioned previously, the use of statistical leverage scores has a long history in statistical and diagnostic regression analysis [49, 50, 51, 110, 189]. To gain insight into these statistical leverage scores, consider the so-called "wood beam data" example [67, 110], which is visually presented in Figure 6.1(a), along with the best-fit line to that data. In Figure 6.1(b), the leverage scores for these ten data points are shown. Intuitively, data points that "stick out" have particularly high leverage — e.g., the data point that has the most influence or leverage on the best-fit line to the wood beam data is the point marked "4", and this is reflected in the relative magnitude of the corresponding statistical leverage score. (Note that the point "1" exhibits some similar behavior; and that although "3" and "9" don't "stick out" in the same sense, they are at the "ends" of the data and possess a relatively-high



(a)

1	0.4179
2	0.2419
3	0.4173
4	0.6044
5	0.2522
6	0.1479
7	0.2616
8	0.1540
9	0.3155
10	0.1873

(b)

Fig. 6.1. Statistical leverage scores historically in diagnostic regression analysis. (a) The Wood Beam Data described in [110] is an example illustrating the use of statistical leverage scores in the context of least-squares approximation. Shown are the original data and the best least-squares fit. (b) The leverage scores for each of the ten data points in the Wood Beam Data. Note that the data point marked "4" has the largest leverage score, as might be expected from visual inspection.

leverage for that reason.) Indeed, since $\mathrm{Trace}(H) = n$, where H is the hat matrix defined in Section 4.1, a rule of thumb that has been suggested in diagnostic regression analysis to identify errors and outliers in a data set is to investigate the i^{th} data point if $H_{ii} > 2n/m$ [51, 189], i.e., if H_{ii} is larger that 2 or 3 times the "average" size. On the other hand, of course, if it happens to turn out that such a point is a legitimate data point, then one might expect that such an outlying data point will be a particularly important or informative data point.

That leverage scores "should be" fairly uniform — indeed, typical conditions even in recent work on the so-called coherence of matrices [9, 40, 48, 183] make just such an assumption — is supported by the idea that if they are not then a small number of data points might be particularly important, in which case a different or more refined statistical model might be appropriate. Furthermore, they are fairly uniform in various limiting cases where measure concentration occurs, e.g., for not-extremely-sparse random graphs, and for matrices such as Laplacians associated with well-shaped low-dimensional manifolds, basically since eigenfunctions tend to be delocalized in those situations. Of course, their actual behavior in realistic data applications is an empirical question.

6.2 More Recent Perspectives on Statistical Leverage

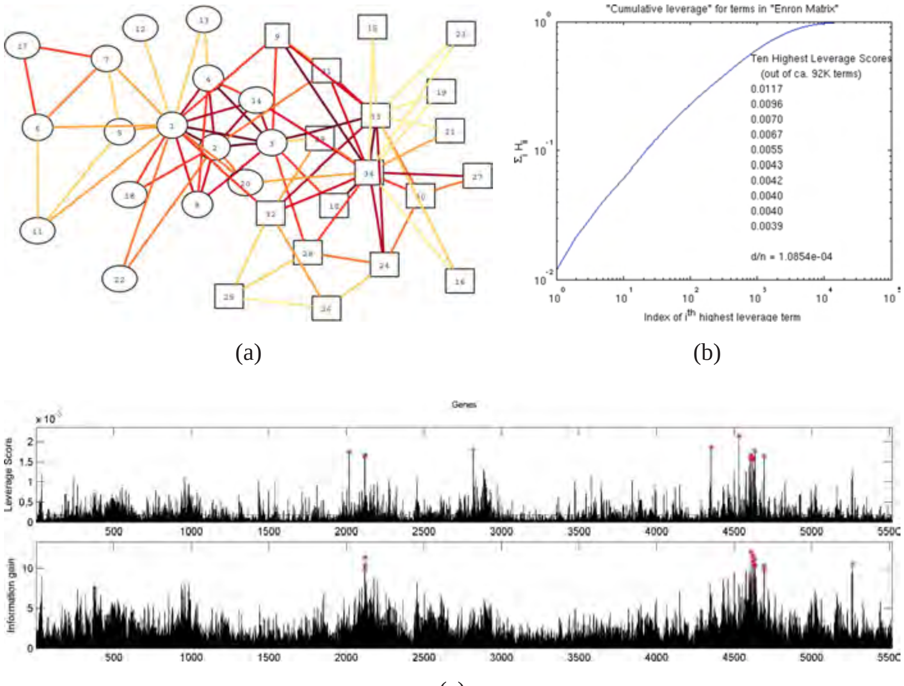
To gain intuition for the behavior of the statistical leverage scores in a typical application, consider Figure 6.2(a), which illustrates the so-called Zachary karate club network [196], a small but popular network in the community detection literature. Given such a network $G = (V,E)$, with n nodes, m edges, and corresponding edge weights $w_e \geq 0$, define the $n \times n$ Laplacian matrix as $L = B^TWB$, where B is the $m \times n$ edge-incidence matrix and W is the $m \times m$ diagonal weight matrix. The effective resistance between two vertices is given by the diagonal entries of the matrix $R = BL^\dagger B^T$ (where $L^\dagger$ denotes the Moore-Penrose generalized inverse) and is related to notions of "network betweenness" [152]. For many large graphs, this and related betweenness measures tend to be strongly correlated with node degree and tend to be large for edges that form articulation points between clusters and communities, i.e., for edges that "stick out" a lot. It can be shown that the effective resistances of the edges of G are proportional to the statistical leverage scores of the m rows of the $m \times n$ matrix $W^{1/2}B$ — consider the $m \times m$ matrix

$$P = W^{1/2}RW^{1/2} = \Phi(\Phi^T\Phi)^\dagger\Phi^T,$$

where $\Phi = W^{1/2}B$, and note that if U_Φ denotes any orthogonal matrix spanning the column space of Φ, then

$$P_{ii} = (U_\Phi U_\Phi^T)_{ii} = ||(U_\Phi)_{(i)}||_2^2.$$

Figure 6.2(a) presents a color-coded illustration of these scores for Zachary karate club network. Note that the higher-leverage red edges tend to be those associated with higher-degree nodes and those at the articulation point between the two clusters.



(a)

(b)

(c)

Fig. 6.2. Statistical leverage scores in more modern applications. (a) The so-called Zachary karate club network [196], with edges color-coded such that leverage scores for a given edge increase from yellow to red. (b) Cumulative leverage (with $k = 10$) for a $65{,}031 \times 92{,}133$ term-document matrix constructed Enron electronic mail collection, illustrating that there are a large number of data points with very high leverage score. (c) The normalized statistical leverage scores and information gain score — information gain is a mutual information-based metric popular in the application area [164, 137] — for each of the $n = 5520$ genes, a situation in which the data cluster well in the low-dimensional space defined by the maximum variance axes of the data [137]. Red stars indicate the 12 genes with the highest leverage scores, and the red dashed line indicates the average or uniform leverage scores. Note the strong correlation between the unsupervised leverage score metric and the supervised information gain metric.

Next, to gain intuition for the (non-)uniformity properties of statistical leverage scores in a typical application, consider a term-document matrix derived from the publicly-released Enron electronic mail collection [22], which is typical of the type of data set to which SVD-based latent semantic analysis (LSA) methods [63] have been applied. This is a $65{,}031 \times 92{,}133$ matrix, as described in [22], and let us choose the rank parameter as $k = 10$. Figure 6.2(b) plots the cumulative leverage, i.e., the running sum of top t statistical leverage scores, as a function of increasing t. Since $\frac{k}{n} = \frac{10}{92{,}133} \approx 1.0854 \times 10^{-4}$, we see that the highest leverage term has a leverage score nearly two orders of magnitude larger than this "average" size scale, that the second highest-leverage score is only marginally less than the first, that the third highest score is marginally less than the second, etc. Thus, by the traditional metrics of diagnostic data analysis [189, 51], which suggests flagging a data point if

$$(P_{U_k})_{ii} = (H_k)_{ii} > 2k/n,$$

there are a *huge* number of data points that are *extremely* outlying, i.e., that are extreme outliers by the metrics of traditional regression diagnostics. In retrospect, of course, this might not be surprising since the Enron email corpus is extremely sparse, with nowhere on the order of $\Omega(n)$ nonzeros per row. Thus, even though LSA methods have been successfully applied, plausible generative models associated with these data are clearly not Gaussian, and the sparsity structure is such that there is no reason to expect that nice phenomena such as measure concentration occur.

Finally, note that DNA microarray and DNA SNP data often exhibit a similar degree of non-uniformity, although for somewhat different reasons. To illustrate, Figure 6.2(c) presents two plots for a data matrix, as was described in [137], consisting of $m = 31$ patients with 3 different cancer types with respect to $n = 5520$ genes. First, this figure plots the information gain — information gain is a mutual information-based metric popular in that application area [164, 137] — for each of the $n = 5520$ genes; and second, it plots the normalized statistical leverage scores for each of these genes. In each case, red dots indicate the genes with the highest values. A similar plot illustrating the remarkable non-uniformity in statistical leverage scores for DNA SNP data was presented in [164]. Empirical evidence suggests that two phenomena may be responsible for this non-uniformity. First, as with the term-document data, there is no domain-specific reason to believe that nice properties like measure concentration occur — on the contrary, there are reasons to expect that they do not. Recall that each DNA SNP corresponds to a single mutational event in human history. Thus, it will "stick out," as its description along its one axis in the vector space will likely not be well-expressed in terms of the other axes, i.e., in terms of the other SNPs, and by the time it "works its way back" due to population admixing, etc., other SNPs will have occurred elsewhere. Second, the correlation between statistical leverage and supervised mutual information-based metrics is particularly prominent in examples where the data cluster well in the low-dimensional space defined by the maximum variance axes. Considering such data sets is, of course, a strong selection bias, but it is common in applications. It would be of interest to develop a model that quantifies the observation that, conditioned on clustering well in the low-dimensional space, an unsupervised measure like leverage scores should be expected to correlate well with a supervised measure like informativeness [164] or information gain [137].

6.3 Statistical Leverage and Selecting Columns from a Matrix

With respect to some of the more technical and implementational issues underlying the CSSP algorithm of Section 5, recall that an important aspect of QR algorithms is how they make so-called pivot rule decisions about which columns to keep [98] and that such decisions can be tricky when the columns are not orthogonal or spread out in similarly nice ways. Several empirical observations [30, 31] are particularly relevant for large-scale data applications.

- We looked at several versions of the QR algorithm, and we compared each version of QR to the CSSP using that version of QR in the second phase. One observation we made was that different QR algorithms behave differently — e.g., some versions such as the Low-RRQR algorithm of [45] tend to perform much better than other versions such as the qrxp algorithm of [26, 27]. Although not surprising to NLA practitioners, this observation indicates that some care should be paid to using "off the shelf" implementations in large-scale applications. A second less-obvious observation is that preprocessing with the randomized first phase tends to improve more poorly-performing variants of QR more than better variants. Part of this is simply that the more poorly-performing variants have more room to improve, but part of this is also that more sophisticated versions of QR tend to make more sophisticated pivot rule decisions, which are relatively less important after the randomized bias toward directions that are "spread out."

- We also looked at selecting columns by applying QR on V_k^T and then keeping the corresponding columns of A, i.e., just running the classical deterministic QR algorithm with no randomized first phase on the matrix V_k^T. Interestingly, with this "preprocessing" we tended to get better columns than if we ran QR on the original matrix A. Again, the interpretation seems to be that, since the norms of the columns of V_k^T define the relevant non-uniformity structure with which to sample with respect to, working directly with those columns tends make things "spread out," thereby avoiding (even in traditional deterministic settings) situations where pivot rules have problems.
- Of course, we also observed that randomization further improves the results, assuming that care is taken in choosing the rank parameter k and the sampling parameter c. In practice, the choice of k should be viewed as a "model selection" question. Then, by choosing $c = k, 1.5k, 2k, \ldots$, we often observed a "sweet spot," in bias-variance sense, as a function of increasing c. That is, for a fixed k, the behavior of the deterministic QR algorithms improves by choosing somewhat more than k columns, but that improvement is degraded by choosing too many columns in the randomized phase.

These and related observations [30, 31] shed light on the inner workings of the CSSP algorithm, the effect of providing a randomized bias toward high-leverage data points at the two stages of the algorithm, and potential directions for the usefulness of this type of randomized algorithm in very large-scale data applications.

6.4 Statistical Leverage in Large-scale Data Analysis

Returning to the genetics applications where the algorithms described in this review have already been applied, we will consider one example each of the two common reasons (faster algorithms and more-interpretable algorithms) described in Section 2.3 for using randomization in the design of matrix algorithms for large-scale data problems. To start with the former motivation, [93] applies the algorithm of [168] that was described in Section 5.3.3 to problems of subspace estimation and prediction in case-control studies in genetics. In particular, [93] extends Principal Component Regression, Sliced Inverse Regression, and Localized Sliced Inverse Regression, three statistical techniques for matrix-based data that use as a critical step a matrix eigendecomposition, to much larger-scale data by using randomization to compute an approximately-optimal basis. Three goals were of interest: evaluate the ability of randomized matrix algorithms to provide a good approximation to the dimension-reduced subspace; evaluate their relative predictive performance, when compared to the exact methods; and evaluate their ability to provide useful graphical summaries for the domain experts.

As an example of their results, Figure 6.3 illustrates pictorially the ability of the randomized algorithm to recover a good approximation to the top principal components of the DNA SNP data of [154]. For appropriate parameter choices, the empirical correlations of the exact sample principal components with the approximate estimates are (starting from second principal component): 0.999, −0.996, 0.930, 0.523, 0.420, 0.255, etc. Thus, the first few components are extremely-well reproduced; and then it appears as if there is some "component splitting," as would be expected from standard matrix perturbation theory, as the seventh principal component from the randomized algorithm correlates relatively-well with three of the exact components. The performance of the randomized algorithm in a Crohn's disease application on a subset of the DNA SNP data from the Wellcome Trust Case Control Consortium [187] illustrates the run-time advantages of exploiting randomization in a real case-control application. For example, when applied to a

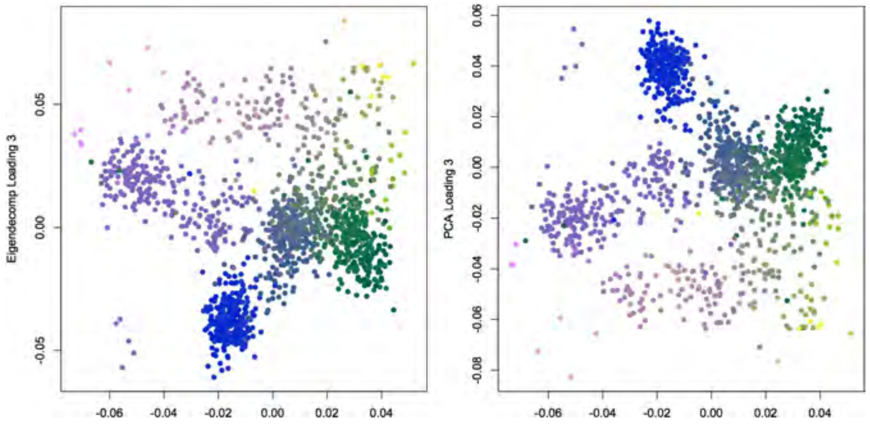


Fig. 6.3. Pictorial illustration of the method of [93] recovering a good approximation to the top principal components of the DNA SNP data of [154]. Left panel is with the randomized algorithm; and right panel is with an exact computation. See the text and [93] for details.

$4,686 \times 6,041$ matrix of SNPs from one chromosome, a single iteration of the randomized algorithm took 6 seconds, versus 37 minutes for the exact deterministic computation (a call to the DGESVD routine in the Lapack package), and it achieved a distance of 0.01 to the exact subspace. By iterating just a few additional times, this distance could be decreased to less than 10^{-6} with a nominal increase in running time relative to the exact computation. Similar results were achieved with matrices of larger sizes, up to a $4,686 \times 29,406$ matrix consisting of SNP data from four chromosomes.

In a somewhat different genetics application, [164] was interested in obtaining a small number of actual DNA SNPs that could be typed and used for ancestry inference and the study of population structure within and across continents around the world. As an example of their results, Figure 6.4 illustrates pictorially the clustering of individuals from nine indigenous populations typed for 9,160 SNPs. k-means clustering was run on the detected significant eigenvectors, and it managed successfully to assign each individual to their country of origin. In order to determine the possibility of identifying a small set of actual SNPs to reproduce this clustering structure, [164] used the statistical leverage scores as a ranking function and selected sets of 10 to 400 actual SNPs and repeated the analysis. Figure 6.4 also illustrates the

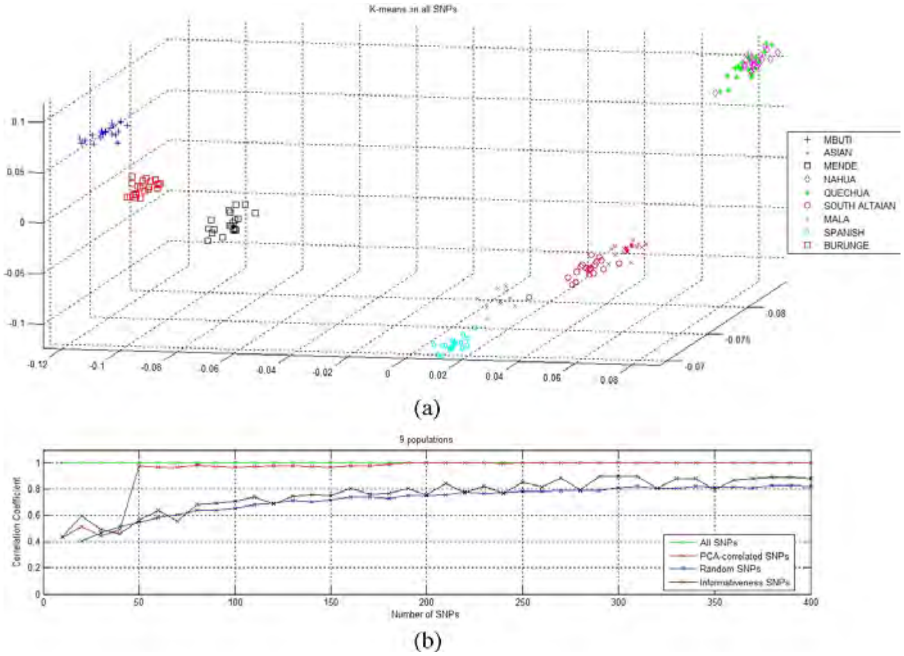


Fig. 6.4. Pictorial illustration of the clustering of individuals from nine populations typed for 9,160 SNPs, from [164]. Top panel illustrates k-means clustering on the full data set. Bottom panel plots, as a function of the number of actual SNPs chosen and for three different SNP selection procedures, the correlation coefficient between the true and predicted membership of the individuals in the nine populations. See the text and [164] for details.

correlation coefficient between the true and predicted membership of the individuals in the nine populations, when the SNPs are chosen in this manner, as well as when the SNPs are chosen with two other procedures (uniformly at random and according to a mutual information-based measure popular in the field). Surprisingly, by using only 50 such "PCA-correlated" SNPs, all individuals were correctly assigned to one of the nine studied populations, as illustrated in the bottom panel of Figure 6.4. (Interestingly, that panel also illustrates that, in this case, the mutual information-based measure performed much worse at selecting SNPs that could then be used to cluster individuals.)

At root, statistical leverage provides one way to quantify the notion of "eigenvector localization," i.e., the idea that most of the mass of an eigenvector is concentrated on a small number of coordinates. This notion arises (often indirectly) in a wide range of scientific computing and data analysis applications; and in *some* of those cases the localization can be meaningfully interpreted in terms of the domain from which the data are drawn. To conclude this section, we will briefly discuss these issues more generally, with an eye toward forward-looking applications.

When working with networks or graph-based data, the so-called *network values* are the eigenvector components associated with the largest eigenvalue of the graph adjacency matrix. In many of these applications, the network values exhibits very high variability; and thus they have been used in a number of contexts [43], most notably to measure the value or worth of a customer for consumer-based applications such as viral marketing [166]. Relatedly, sociologists have used so-called *centrality* measures to measure the importance of individual nodes in a network. The most relevant centrality notions for us are the Bonacich centrality [28] and the random walk centrality of Newman [152], both of which are variants of these network values. In still other network applications, effective resistances (recall the connection discussed in Section 6.2) have been used to characterize distributed control and estimation problems [15] as well as problems of asymptotic space localization in sensor networks [117].

In many scientific applications, localized eigenvectors have a very natural interpretation — after all, physical particles (as well as photons, phonons, etc.) themselves are localized eigenstates of appropriate Hamiltonian operators. If the so-called density matrix of a physical system is given by $\rho(r,r^{'}) = \sum_{i=1}^{N} \psi_i(r)^T \psi_i(r^{'})$, then if V is the matrix whose column vectors are the normalized eigenvectors $\psi_i, i = 1,\ldots,s$, for the s occupied states, then $P = VV^T$ is a projection matrix, and the charge density at a point r_i in space is the i^{th} diagonal element of P. (Here the transpose actually refers to the Hermitian conjugate since the wavefunction $\psi_i(r)$ is a complex quantity.) Thus, the magnitude of this entry as well as other derived quantities like the trace of ρ give empirically-measurable quantities; see, e.g., Section 6.2 of [173]. More practically, improved methods for estimating the diagonal of a projection matrix may have significant implications for leading to improvements in large-scale numerical computations in scientific computing applications such as the density functional theory of many-atom systems [16, 173].

In other physical applications, localized eigenvectors arise when extreme sparsity is coupled with randomness or quasi-randomness. For example, [34, 167] describe a model for diffusion in a configuration space that combines features of infinite dimensionality and very low connectivity — for readers more familiar with the Erdős-Rényi G_{np} random graph model [80] than with spin glass theory, the parameter region of interest in these applications corresponds to the extremely sparse region $1/n \lesssim p \lesssim \log n/n$. For ensembles of very sparse random matrices, there is a localization-delocalization transition which has been studied in detail [82, 92, 122, 148]. In these applications, as a rule of thumb, eigenvector localization occurs when there is some sort of "structural heterogeneity," e.g., the degree (or coordination number) of a node is significantly higher or lower than average.

Many large complex networks that have been popular in recent years, e.g., social and information networks, networks constructed from biological data, networks constructed from financial transactions, etc., exhibit similar qualitative properties, largely since these networks are often very sparse and relatively-unstructured at large size scales. See, e.g., [66, 83, 96, 149] for detailed discussions and empirical evaluations. Depending on whether one is considering the adjacency matrix or the Laplacian matrix, localized eigenvectors can correspond to structural inhomogeneities such as very high degree nodes or very small cluster-like sets of nodes. In addition, localization is often preserved or modified in characteristic ways when a graph is generated by modifying an existing graph in a structured manner; and thus it has been used as a diagnostic in certain network applications [13, 14]. The implications of the algorithms described in this review remain to be explored for these and other applications where eigenvector localization is a significant phenomenon.

7

A Few General Thoughts, and A Few Lessons Learned

7.1 Thoughts about Statistical Leverage in MMDS Applications

One high-level question raised by the theoretical and empirical results reviewed here is:

- Why are the statistical leverage scores so nonuniform in many large-scale data analysis applications?

The answer to this seems to be that, intuitively, in many MMDS application areas, statistical models are *implicitly* assumed based on computational and not statistical considerations. That is, when computational decisions are made, often with little regard for the statistical properties of the data, they carry with them statistical consequences, in the sense that the computations are the "right thing" or the "wrong thing" to do for different classes of data. Thus, in these cases, it is not surprising that some interesting data points "stick out" relative to obviously inappropriate models. This suggests the use of these importance sampling scores as cheap signatures of the "inappropriateness" of a statistical model (chosen for algorithmic and not statistical reasons) in large-scale exploratory or diagnostic applications.

A second high-level question raised by the results reviewed here is:

- Why should statistical leverage, a traditional concept from regression diagnostics, be useful to obtain improved worst-case approximation algorithms for traditional NLA matrix problems?

Here, the answer seems to be that, intuitively, if a data point has a high leverage score and is not an error then it might be a particularly important or informative data point. Since worst-case analysis takes the input matrix as given, each row/column is assumed to be reliable, and so worst-case guarantees are obtained by focusing effort on the most informative data points. It would be interesting to see if this perspective is applicable more generally in the design of matrix and graph algorithms for other MMDS applications.

7.2 Lessons Learned about Transferring Theory to Practice

More generally, it should be emphasized that the randomized matrix algorithms reviewed here represent a real success story in bringing novel theoretical ideas to the solution of practical problems. This often-cited (but less frequently-achieved) goal arises in many MMDS applications — in fact, bridging the gap between TCS, NLA, and data applications was at the origin of the MMDS meetings [97, 136, 138], which address algorithmic and statistical aspects of large-scale data analysis more generally. Not surprisingly, the widespread interest in these algorithms is not simply due to the strong worst-case bounds they achieve, but also to their usefulness in downstream data analysis and scientific computing applications. Thus, it is worth highlighting some of the "lessons learned" that are applicable more generally than to the particular algorithms and applications reviewed here.

- **Objective functions versus certificates.** TCS is typically concerned with providing bounds on objective functions in approximate optimization problems (as in, e.g., (4.2) and (5.1)) and makes no statement about how close the certificate (i.e., the vector or graph achieving that approximate solution) is to a/the exact solution of the optimization problem (as in, e.g., (4.3)). In machine learning and data analysis, on the other hand, one is often interested in statements about the quality of the certificate, largely since the certificate is often used more generally for other downstream applications like clustering or classification.

- **Identifying structure versus washing out structure.** TCS is often *not* interested in identifying structure *per se*, but instead only in exploiting that structure to provide fast algorithms. Thus, important structural statements are often buried deep in the analysis of the algorithm. Making such structural statements explicit has several benefits: one can obtain improved bounds if the tools are more powerful than originally realized (as when relative-error projection algorithms followed additive-error projection algorithms and relative-error sampling algorithms simply by performing a more sophisticated analysis); structural properties can be of independent interest in downstream data applications; and it can make it easer to couple to more traditional numerical methods.
- **Side effects of computational decisions.** There are often side effects of computational decisions that are at least as important for the success of novel methods as is the original nominal reason for the introduction of the new methods. For example, randomness was originally used as a resource inside the algorithm to speed up the running time of algorithms on worst-case input. On the other hand, using randomness inside the algorithm often leads to improved condition number properties, better parallelism properties on modern computational architectures, and better implicit regularization properties, in which case the approximate answer can be even better than the exact answer for downstream applications.
- **Significance of cultural issues.** TCS would say that if a randomized algorithm succeeds with constant probability, say with probability at least 90%, then it can be boosted to hold with probability at least $1 - \delta$, where the dependence on

δ scales as $O(\log(1/\delta))$, using standard methods [150]. Some areas would simply say that such an algorithm succeeds with "overwhelming probability" or fails with "negligible probability." Still other areas like NLA and scientific computing are more willing to embrace randomness if the constants are folded into the algorithm such that the algorithm fails with probability less than, say, 10^{-17}. Perhaps surprisingly, getting beyond such seemingly-minor cultural differences has been the main bottleneck to technology transfer such as that reviewed here.

- **Coupling with domain experience.** Since new methods almost always perform more poorly than well-established methods on traditional metrics, a lot can be gained by coupling with domain expertise and traditional machinery. For example, by coupling with traditional iterative methods, minor variants of the original randomized algorithms for the LS problem can have their ϵ dependence improved from roughly $O(1/\epsilon)$ to $O(\log(1/\epsilon))$. Similarly, since factors of 2 matter for geneticists, by using the leverage scores as a ranking function rather than as an importance sampling distribution, greedily keeping, say, 100 SNPs and then filtering to 50 according to a genetic criterion, one often does very well in those applications.

8

Conclusion

Randomization has had a long history in scientific applications [108, 147]. For example, originally developed to evaluate phase space integrals in liquid-state statistical mechanics, Markov chain Monte Carlo techniques are now widely-used in applications as diverse as option valuation in finance, drug design in computational chemistry, and Bayesian inference in statistics. Similarly, originally developed to describe the energy levels of systems arising in nuclear physics, random matrix theory has found applications in areas as diverse as signal processing, finance, multivariate statistics, and number theory. Randomized methods have been popular in these and other scientific applications for several reasons: the weakness of the assumptions underlying the method permits its broad applicability; the simplicity of these assumptions has permitted a rich body of theoretical work that has fruitfully fed back into applications; due to the intuitive connection between the method and hypothesized noise properties in the data; and since randomization permits the approximate solution of otherwise impossible-to-solve problems.

Within the last few decades, randomization has also proven to be useful in a very different way — as a powerful resource in TCS for establishing worst-case bounds for a wide range of computational problems. That is, in the same way that space and time are valuable resources available to be used judiciously by algorithms, it has been discovered that exploiting randomness as an algorithmic resource *inside the algorithm* can lead to better algorithms. Here, "better" typically means faster in worst-case theory when compared, e.g., to deterministic algorithms for the same problem; but it could also mean simpler — which is of typically interest since simpler algorithms tend to be more amenable to worst-case theoretical analysis. Applications of this paradigm have included algorithms for number theoretic problems such as primality testing, algorithms for data structure problems such as sorting and order statistics, as well as algorithms for a wide range of optimization and graph theoretic problems such as linear programming, minimum spanning trees, shortest paths, and minimum cuts.

Perhaps since its original promise was oversold, and perhaps due to the greater-than-expected difficulty in developing high-quality numerically-stable software for scientific computing applications, randomization *inside the algorithm* for common matrix problems was mostly "banished" from scientific computing and NLA in the 1950s. Thus, it is refreshing that within just the last few years, novel algorithmic perspectives from TCS have worked their way back to NLA, scientific computing, and scientific data analysis. These developments have been driven by large-scale data analysis applications, which place very different demands on matrices than traditional scientific computing applications. As with other applications of randomization, though, the ideas underlying these developments are simple, powerful, and broadly-applicable.

Several obvious future directions seem particularly promising application areas for this randomized matrix algorithm paradigm.

- **Other traditional NLA problems and large-scale optimization.** Although least squares approximation and low-rank matrix approximation are fundamental problems that underlie a wide range of problems, there are many other problems of interest in NLA — computing polar decompositions, eigenvalue decompositions, Cholesky decompositions, etc.

In addition, large-scale numerical optimization code often uses these primitives many times during the course of a single computation. Thus, for example, some of the fast numerical implementations for very overdetermined least squares problems that were described in Section 4.5 can in principle be used to accelerate interior-point methods for convex optimization and linear programming. Working through the practice in realistic computational settings remains an ongoing challenge.

- **Parallel and distributed computational environments.** In many applications, communication is more expensive than computation. This is true both for computations involving a single machine — recall recent developments in multicore computing — as well as for computations run across many machines — such as in large distributed data centers. In some cases, such as with Gaussian-based random projections, computations can be easily parallelized; and numerical implementations for both the least squares approximation problem and the low-rank approximation problem have already exploited this. Taking advantage of modern computer architectures and systems requirements more generally is a substantial challenge.
- **Sparse graphs, sparse matrices, and sparse projections.** Sparsity is a ubiquitous property of data, and one which is a challenge since vanilla applications of randomized algorithms tend to densify the input data. In some cases, sparsity in the input is structured and can be exploited by the randomized algorithm; while in other cases it is less structured but it can be respected with existing projection methods. More generally, sparse projection matrices are of interest — such projection matrices make it easy to perform incremental updates in data streaming environments, they can make it easier to perform matrix-vector products quickly, etc. Problems of recovering sparse signals have been approached by researchers in theoretical computer science, applied mathematics, and digital signal processing; and in

many cases the approaches are somewhat orthogonal to that of the work reviewed here.

- **Laplacian matrices and large informatics graphs.** Laplacians are fundamental matrices associated with a graph, and they permit many of the randomized matrix algorithms we have been discussing to be applied to graph-based data. In some cases, the goal might be to sparsify an input graph; but more typically graphs arising in data applications are sparse and irregular. In the case of large social and information networks, for example, it is known that, while often there exists good small-scale clustering structure, there typically does not exist good large-scale clustering structure. Part of this has to do with the heavy-tailed properties in these graphs, which imply that although there may exist a small number of most important nodes in the graph, these nodes do not capture most of the information in the data. This presents numerous fundamental challenges for the algorithms reviewed here, and these challenges have just begun to be addressed.
- **Randomized algorithms and implicit regularization.** In many cases, randomized algorithms or their output are more robust than their deterministic variants. For example, algorithms may empirically be less sensitive to pivot rule decisions; and their output may empirically be "nicer" and more "regular" — in the sense of statistical regularization. Existing theory (reviewed here) makes precise a sense in which randomized matrix algorithms are not much worse than the corresponding deterministic algorithms; but quantifying a sense in which the output of randomized matrix algorithms is even "better" than the output of the corresponding deterministic algorithms is clearly of interest if one is interested in very large-scale applications.

In closing, it seems worth reminding the reader that researchers often look with great expectation toward randomness, as if a naïve application of randomness will somehow solve all of one's problems. By now, it should be clear that such hopes are rarely realized in practice. It should also be clear, though, that a careful application of randomness — especially when it is coupled closely with domain expertise — provides a powerful framework to address a range of matrix-based problems in modern massive data set analysis.

Acknowledgments

I would like to thank the numerous colleagues and collaborators with whom these results have been discussed in preliminary form — in particular, Petros Drineas, with whom my contribution to the work reviewed here was made; Michael Jordan and Deepak Agarwal, who first pointed out the connection between the importance sampling probabilities used in the relative-error matrix approximation algorithms and the concept of statistical leverage; Sayan Mukherjee, who generously provided unpublished results on the usefulness of randomized matrix algorithms in his genetics applications; and Ameet Talwalkar and an anonymous reviewer for providing numerous valuable comments.

References

[1] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *Journal of Computer and System Sciences*, vol. 66, no. 4, pp. 671–687, 2003.

[2] D. Achlioptas and F. McSherry, "Fast computation of low-rank matrix approximations," *Journal of the ACM*, vol. 54, no. 2, p. Article 9, 2007.

[3] N. Ailon and B. Chazelle, "Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform," in *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pp. 557–563, 2006.

[4] N. Ailon and B. Chazelle, "The fast Johnson-Lindenstrauss transform and approximate nearest neighbors," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 302–322, 2009.

[5] N. Ailon and B. Chazelle, "Faster dimension reduction," *Communications of the ACM*, vol. 53, no. 2, pp. 97–104, 2010.

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

[7] N. Ailon and E. Liberty, "An almost optimal unrestricted fast Johnson-Lindenstrauss transform," in *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 185–191, 2011.

[8] O. Alter, P. O. Brown, and D. Botstein, "Singular value decomposition for genome-wide expression data processing and modeling," *Proc. Natl. Acad. Sci. USA*, vol. 97, no. 18, pp. 10101–10106, 2000.

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

[10] M. Baboulin, J. Dongarra, and S. Tomov, "Some issues in dense linear algebra for multicore and special purpose architectures," Technical Report UT-CS-08-200, University of Tennessee, May 2008.

[11] N. M. Ball and R. J. Brunner, "Data mining and machine learning in astronomy," *International Journal of Modern Physics D*, vol. 19, no. 7, pp. 1049–1106, 2010.

[12] N. M. Ball, J. Loveday, M. Fukugita, O. Nakamura, S. Okamura, J. Brinkmann, and R. J. Brunner, "Galaxy types in the Sloan Digital Sky Survey using supervised artificial neural networks," *Monthly Notices of the Royal Astronomical Society*, vol. 348, no. 3, pp. 1038–1046, 2004.

[13] A. Banerjee and J. Jost, "On the spectrum of the normalized graph Laplacian," *Linear Algebra and its Applications*, vol. 428, no. 11–12, pp. 3015–3022, 2008.

[14] A. Banerjee and J. Jost, "Graph spectra as a systematic tool in computational biology," *Discrete Applied Mathematics*, vol. 157, no. 10, pp. 2425–2431, 2009.

[15] P. Barooah and J. P. Hespanha, "Graph effective resistances and distributed control: Spectral properties and applications," in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 3479–3485, 2006.

[16] C. Bekas, E. Kokiopoulou, and Y. Saad, "An estimator for the diagonal of a matrix," *Applied Numerical Mathematics*, vol. 57, pp. 1214–1229, 2007.

[17] M.-A. Belabbas and P. J. Wolfe, "Fast low-rank approximation for covariance matrices," in *Second IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 293–296, 2007.

[18] M.-A. Belabbas and P. J. Wolfe, "On sparse representations of linear operators and the approximation of matrix products," in *Proceedings of the 42nd Annual Conference on Information Sciences and Systems*, pp. 258–263, 2008.

[19] M.-A. Belabbas and P. J. Wolfe, "On landmark selection and sampling in high-dimensional data analysis," *Philosophical Transactions of the Royal Society, Series A*, vol. 367, pp. 4295–4312, 2009.

[20] M.-A. Belabbas and P. J. Wolfe, "Spectral methods in machine learning and new strategies for very large datasets," *Proc. Natl. Acad. Sci. USA*, vol. 106, pp. 369–374, 2009.

[21] M.-A. Belabbas and P. Wolfe, "On the approximation of matrix products and positive definite matrices," Technical report. Preprint: arXiv:0707.4448, 2007.

[22] M. W. Berry and M. Browne, "Email surveillance using non-negative matrix factorization," *Computational and Mathematical Organization Theory*, vol. 11, no. 3, pp. 249–264, 2005.

[23] 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.

[24] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proceedings of the 7th Annual ACM SIGKDD Conference*, pp. 245–250, 2001.

[25] C. H. Bischof and P. C. Hansen, "Structure-preserving and rank-revealing QR-factorizations," *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 6, pp. 1332–1350, 1991.

[26] C. H. Bischof and G. Quintana-Ortí, “Algorithm 782: Codes for rank-revealing QR factorizations of dense matrices,” *ACM Transactions on Mathematical Software*, vol. 24, no. 2, pp. 254–257, 1998.

[27] C. H. Bischof and G. Quintana-Ortí, “Computing rank-revealing QR factorizations of dense matrices,” *ACM Transactions on Mathematical Software*, vol. 24, no. 2, pp. 226–253, 1998.

[28] P. Bonacich, “Power and centrality: A family of measures,” *The American Journal of Sociology*, vol. 92, no. 5, pp. 1170–1182, 1987.

[29] T. A. Boroson and T. R. Lauer, “Exploring the spectral space of low redshift QSOs,” *The Astronomical Journal*, vol. 140, pp. 390–402, 2010.

[30] C. Boutsidis, M. W. Mahoney, and P. Drineas, “An improved approximation algorithm for the column subset selection problem,” Technical report. Preprint: arXiv:0812.4293v2, 2008.

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

[32] C. Boutsidis, M. W. Mahoney, and P. Drineas, “Unsupervised feature selection for the k-means clustering problem,” in *Annual Advances in Neural Information Processing Systems 22: Proceedings of the 2009 Conference*, 2009.

[33] 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*, pp. 968–977, 2009.

[34] A. J. Bray and G. J. Rodgers, “Diffusion in a sparsely connected space: A model for glassy relaxation,” *Physical Review B*, vol. 38, no. 16, pp. 11461–11470, 1988.

[35] M. E. Broadbent, M. Brown, and K. Penner, “Subset selection algorithms: Randomized vs. deterministic,” *SIAM Undergraduate Research Online*, vol. 3, May 13 2010.

[36] R. J. Brunner, S. G. Djorgovski, T. A. Prince, and A. S. Szalay, “Massive datasets in astronomy,” in *Handbook of Massive Data Sets*, (J. Abello, P. M. Pardalos, and M. G. C. Resende, eds.), pp. 931–979, Kluwer Academic Publishers, 2002.

[37] T. Budavári, V. Wild, A. S. Szalay, L. Dobos, and C.-W. Yip, “Reliable eigenspectra for new generation surveys,” *Monthly Notices of the Royal Astronomical Society*, vol. 394, no. 3, pp. 1496–1502, 2009.

[38] P. Businger and G. H. Golub, “Linear least squares solutions by Householder transformations,” *Numerische Mathematik*, vol. 7, pp. 269–276, 1965.

[39] E. Candes, L. Demanet, and L. Ying, “Fast computation of Fourier integral operators,” *SIAM Journal on Scientific Computing*, vol. 29, no. 6, pp. 2464–2493, 2007.

[40] E. Candes and J. Romberg, “Sparsity and incoherence in compressive sampling,” *Inverse Problems*, vol. 23, no. 3, pp. 969–985, 2007.

[41] E. J. Candes and B. Recht, “Exact matrix completion via convex optimization,” *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[42] S. Chaillat and G. Biros, "FaIMS: A fast algorithm for the inverse medium problem with multiple frequencies and multiple sources for the scalar Helmholtz equation," Manuscript, 2010.

[43] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Computing Surveys*, vol. 38, no. 1, p. 2, 2006.

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

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

[46] T. F. Chan and P. C. Hansen, "Computing truncated singular value decomposition least squares solutions by rank revealing QR-factorizations," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, pp. 519–530, 1990.

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

[48] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, no. 2, pp. 572–596, 2011.

[49] S. Chatterjee and A. S. Hadi, "Influential observations, high leverage points, and outliers in linear regression," *Statistical Science*, vol. 1, no. 3, pp. 379–393, 1986.

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

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

[52] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Molecular Cell*, vol. 2, pp. 65–73, 1998.

[53] A. Civril and M. Magdon-Ismail, "Deterministic sparse column based matrix reconstruction via greedy approximation of SVD," in *Proceedings of the 19th Annual International Symposium on Algorithms and Computation*, pp. 414–423, 2008.

[54] A. Civril and M. Magdon-Ismail, "On selecting a maximum volume sub-matrix of a matrix and related problems," *Theoretical Computer Science*, vol. 410, pp. 4801–4811, 2009.

[55] K. L. Clarkson and D. P. Woodruff, "Numerical linear algebra in the streaming model," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pp. 205–214, 2009.

[56] E. S. Coakley, V. Rokhlin, and M. Tygert, "A fast randomized algorithm for orthogonal projection," *SIAM Journal on Scientific Computing*, vol. 33, no. 2, pp. 849–868, 2011.

[57] A. J. Connolly and A. S. Szalay, "A robust classification of galaxy spectra: Dealing with noisy and incomplete data," *The Astronomical Journal*, vol. 117, no. 5, pp. 2052–2062, 1999.

[58] A. J. Connolly, A. S. Szalay, M. A. Bershady, A. L. Kinney, and D. Calzetti, "Spectral classification of galaxies: an orthogonal approach," *The Astronomical Journal*, vol. 110, no. 3, pp. 1071–1082, 1995.

[59] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[60] A. Dasgupta, R. Kumar, and T. Sarlós, "A sparse Johnson-Lindenstrauss transform," in *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pp. 341–350, 2010.

[61] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss," *Random Structures and Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[62] A. d'Aspremont, "Subsampling algorithms for semidefinite programming," Technical Report. Preprint: arXiv:0803.1990, 2008.

[63] S. T. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[64] A. Deshpande and S. Vempala, "Adaptive sampling and fast low-rank matrix approximation," Technical Report TR06-042, Electronic Colloquium on Computational Complexity, March 2006.

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

[66] S. N. Dorogovtsev, A. V. Goltsev, J. F. F. Mendes, and A. N. Samukhin, "Spectra of complex networks," *Physical Review E*, vol. 68, p. 046109, 2003.

[67] N. R. Draper and D. M. Stoneman, "Testing for the inclusion of variables in linear regression by a randomisation technique," *Technometrics*, vol. 8, no. 4, pp. 695–699, 1966.

[68] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, "Clustering large graphs via the singular value decomposition," *Machine Learning*, vol. 56, no. 1–3, pp. 9–33, 2004.

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

[70] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix," *SIAM Journal on Computing*, vol. 36, pp. 158–183, 2006.

[71] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition," *SIAM Journal on Computing*, vol. 36, pp. 184–206, 2006.

[72] P. Drineas, J. Lewis, and P. Paschou, "Inferring geographic coordinates of origin for Europeans using small panels of ancestry informative markers," *PLoS ONE*, vol. 5, no. 8, no. 8, p. e11892, 2010.

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

[74] 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*, vol. 6, pp. 2153–2175, 2005.

[75] 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*, pp. 1127–1136, 2006.

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

[77] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, “Faster least squares approximation,” *Numerische Mathematik*, vol. 117, no. 2, pp. 219–249, 2010.

[78] B. Engquist and O. Runborg, “Wavelet-based numerical homogenization with applications,” in *Multiscale and Multiresolution Methods: Theory and Applications*, LNCSE, (T. J. Barth, T. F. Chan, and R. Haimes, eds.), pp. 97–148, Springer, 2001.

[79] B. Engquist and L. Ying, “Fast directional multilevel algorithms for oscillatory kernels,” *SIAM Journal on Scientific Computing*, vol. 29, pp. 1710–1737, 2007.

[80] P. Erdős and A. Rényi, “On the evolution of random graphs,” *Publ. Math. Inst. Hungar. Acad. Sci*, vol. 5, pp. 17–61, 1960.

[81] S. Eriksson-Bique, M. Solbrig, M. Stefanelli, S. Warkentin, R. Abbey, and I. C. F. Ipsen, “Importance sampling for a Monte Carlo matrix multiplication algorithm, with application to information retrieval,” *SIAM Journal on Scientific Computing*, vol. 33, no. 4, pp. 1689–1706, 2011.

[82] S. N. Evangelou, “A numerical study of sparse random matrices,” *Journal of Statistical Physics*, vol. 69, no. 1-2, pp. 361–383, 1992.

[83] I. J. Farkas, I. Derényi, A.-L. Barabási, and T. Vicsek, “Spectra of “real-world” graphs: Beyond the semicircle law,” *Physical Review E*, vol. 64, p. 026704, 2001.

[84] X. Z. Fern and C. E. Brodley, “Random projection for high dimensional data clustering: A cluster ensemble approach,” in *Proceedings of the 20th International Conference on Machine Learning*, pp. 186–193, 2003.

[85] S. R. Folkes, O. Lahav, and S. J. Maddox, “An artificial neural network approach to the classification of galaxy spectra,” *Mon. Not. R. Astron. Soc.*, vol. 283, no. 2, pp. 651–665, 1996.

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

[87] D. Fradkin and D. Madigan, “Experiments with random projections for machine learning,” in *Proceedings of the 9th Annual ACM SIGKDD Conference*, pp. 517–522, 2003.

[88] P. Frankl and H. Maehara, “The Johnson-Lindenstrauss lemma and the sphericity of some graphs,” *Journal of Combinatorial Theory Series A*, vol. 44, no. 3, pp. 355–362, 1987.

[89] A. Frieze and R. Kannan, “Quick approximation to matrices and applications,” *Combinatorica*, vol. 19, no. 2, pp. 175–220, 1999.

[90] A. Frieze, R. Kannan, and S. Vempala, "Fast Monte-Carlo algorithms for finding low-rank approximations," *Journal of the ACM*, vol. 51, no. 6, pp. 1025–1041, 2004.

[91] Z. Füredi and J. Komlós, "The eigenvalues of random symmetric matrices," *Combinatorica*, vol. 1, no. 3, pp. 233–241, 1981.

[92] Y. V. Fyodorov and A. D. Mirlin, "Localization in ensemble of sparse random matrices," *Physical Review Letters*, vol. 67, pp. 2049–2052, 1991.

[93] S. Georgiev and S. Mukherjee, Unpublished results. 2011.

[94] A. Gilbert and P. Indyk, "Sparse recovery using sparse matrices," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 937–947, 2010.

[95] N. Goel, G. Bebis, and A. Nefian, "Face recognition experiments with random projection," *Proceedings of the SPIE*, vol. 5779, pp. 426–437, 2005.

[96] K.-I. Goh, B. Kahng, and D. Kim, "Spectra and eigenvectors of scale-free networks," *Physical Review E*, vol. 64, p. 051903, 2001.

[97] G. H. Golub, M. W. Mahoney, P. Drineas, and L.-H. Lim, "Bridging the gap between numerical linear algebra, theoretical computer science, and data applications," *SIAM News*, vol. 39, no. 8, October 2006.

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

[99] S. A. Goreinov and E. E. Tyrtyshnikov, "The maximum-volume concept in approximation by low-rank matrices," *Contemporary Mathematics*, vol. 280, pp. 47–51, 2001.

[100] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, "A theory of pseudoskeleton approximations," *Linear Algebra and Its Applications*, vol. 261, pp. 1–21, 1997.

[101] S. J. Gould, *The Mismeasure of Man.* New York: W. W. Norton and Company, 1996.

[102] L. Grasedyck and W. Hackbusch, "Construction and arithmetics of H-matrices," *Computing*, vol. 70, no. 4, pp. 295–334, 2003.

[103] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 73, no. 2, pp. 325–348, 1987.

[104] L. Greengard and V. Rokhlin, "A new version of the fast multipole method for the Laplace equation in three dimensions," *Acta Numerica*, vol. 6, pp. 229–269, 1997.

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

[106] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert, "An algorithm for the principal component analysis of large data sets," Technical report. Preprint: arXiv:1007.5510, 2010.

[107] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, no. 2, pp. 217–288, 2011.

[108] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods.* London and New York: Chapman and Hall, 1964.

[109] S. Har-Peled, "Low rank matrix approximation in linear time," Manuscript, January 2006.

[110] D. C. Hoaglin and R. E. Welsch, “The hat matrix in regression and ANOVA,” *The American Statistician*, vol. 32, no. 1, pp. 17–22, 1978.

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

[112] B. D. Horne and N. J. Camp, “Principal component analysis for selection of optimal SNP-sets that capture intragenic genetic variation,” *Genetic Epidemiology*, vol. 26, no. 1, pp. 11–21, 2004.

[113] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 604–613, 1998.

[114] A. Javed, P. Drineas, M. W. Mahoney, and P. Paschou, “Efficient genomewide selection of PCA-correlated tSNPs for genotype imputation,” *Annals of Human Genetics*, vol. 75, no. 6, pp. 707–722, 2011.

[115] W. B. Johnson and J. Lindenstrauss, “Extensions of Lipshitz mapping into Hilbert space,” *Contemporary Mathematics*, vol. 26, pp. 189–206, 1984.

[116] I. M. Johnstone, “On the distribution of the largest eigenvalue in principal components analysis,” *Annals of Statistics*, vol. 29, no. 2, pp. 295–327, 2001.

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

[118] D. M. Kane and J. Nelson, “A derandomized sparse Johnson-Lindenstrauss transform,” Technical Report. Preprint: arXiv:1006.3585, 2010.

[119] D. M. Kane and J. Nelson, “Sparser Johnson-Lindenstrauss transforms,” Technical Report. Preprint: arXiv:1012.1577, 2010.

[120] S. Kaski, “Dimensionality reduction by random mapping: fast similarity computation for clustering,” in *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, pp. 413–418, 1998.

[121] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[122] R. Kühn, “Spectra of sparse random matrices,” *J. of Physics A: Math. and Theor*, vol. 41, p. 295002, 2008.

[123] S. Kumar, M. Mohri, and A. Talwalkar, “On sampling-based approximate spectral decomposition,” in *Proceedings of the 26th International Conference on Machine Learning*, pp. 553–560, 2009.

[124] S. Kumar, M. Mohri, and A. Talwalkar, “Sampling techniques for the Nyström method,” in *Proceedings of the 12th Tenth International Workshop on Artificial Intelligence and Statistics*, pp. 304–311, 2009.

[125] F. G. Kuruvilla, P. J. Park, and S. L. Schreiber, “Vector algebra in the analysis of genome-wide expression data,” *Genome Biology*, vol. 3, no. 3, pp. research0011.1–0011.11, 2002.

[126] M. Li, J. T. Kwok, and B.-L. Lu, “Making large-scale Nyström approximation possible,” in *Proceedings of the 27th International Conference on Machine Learning*, pp. 631–638, 2010.

[127] E. Liberty, N. Ailon, and A. Singer, "Dense fast random projections and lean Walsh transforms," in *Proceedings of the 12th International Workshop on Randomization and Computation*, pp. 512–522, 2008.

[128] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, "Randomized algorithms for the low-rank approximation of matrices," *Proc. Natl. Acad. Sci. USA*, vol. 104, no. 51, pp. 20167–20172, 2007.

[129] L. Lin, J. Lu, and L. Ying, "Fast construction of hierarchical matrix representation from matrix-vector multiplication," *Journal of Computational Physics*, vol. 230, pp. 4071–4087, 2011.

[130] L. Lin, C. Yang, J. C. Meza, J. Lu, L. Ying, and W. E. SelInv, "An algorithm for selected inversion of a sparse symmetric matrix," *ACM Transactions on Mathematical Software*, vol. 37, no. 4, p. 40, 2011.

[131] Z. Lin and R. B. Altman, "Finding haplotype tagging SNPs by use of principal components analysis," *American Journal of Human Genetics*, vol. 75, pp. 850–861, 2004.

[132] L. Mackey, A. Talwalkar, and M. I. Jordan, "Divide-and-conquer matrix factorization," Technical report. Preprint: arXiv:1107.0789, 2011.

[133] D. Madgwick, O. Lahav, K. Taylor, and the 2dFGRS Team, "Parameterisation of galaxy spectra in the 2dF galaxy redshift survey," in *Mining the Sky: Proceedings of the MPA/ESO/MPE Workshop, ESO Astrophysics Symposia*, pp. 331–336, 2001.

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

[135] A. Magen and A. Zouzias, "Low rank matrix-valued Chernoff bounds and approximate matrix multiplication," in *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1422–1436, 2011.

[136] M. W. Mahoney, "Computation in large-scale scientific and Internet data applications is a focus of MMDS 2010," Technical report. Preprint: arXiv:1012.4231, 2010.

[137] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proc. Natl. Acad. Sci. USA*, vol. 106, pp. 697–702, 2009.

[138] M. W. Mahoney, L.-H. Lim, and G. E. Carlsson, "Algorithmic and statistical challenges in modern large-scale data analysis are the focus of MMDS 2008," Technical report. Preprint: arXiv:0812.3702, 2008.

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

[140] P.-G. Martinsson, "Rapid factorization of structured matrices via randomized sampling," Technical Report. Preprint: arXiv:0806.2339, 2008.

[141] P.-G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the decomposition of matrices," *Applied and Computational Harmonic Analysis*, vol. 30, pp. 47–68, 2011.

[142] J. Matousek, "On variants of the Johnson–Lindenstrauss lemma," *Random Structures and Algorithms*, vol. 33, no. 2, pp. 142–156, 2008.

[143] R. C. McGurk, A. E. Kimball, and Z. Ivezić, “Principal component analysis of SDSS stellar spectra,” *The Astronomical Journal*, vol. 139, pp. 1261–1268, 2010.

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

[145] Z. Meng, D. V. Zaykin, C. F. Xu, M. Wagner, and M. G. Ehm, “Selection of genetic markers for association analyses, using linkage disequilibrium and haplotypes,” *American Journal of Human Genetics*, vol. 73, no. 1, pp. 115–130, 2003.

[146] P. Menozzi, A. Piazza, and L. Cavalli-Sforza, “Synthetic maps of human gene frequencies in Europeans,” *Science*, vol. 201, no. 4358, pp. 786–792, 1978.

[147] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[148] A. D. Mirlin and Y. V. Fyodorov, “Universality of level correlation function of sparse random matrices,” *J. Phys. A: Math. Gen*, vol. 24, pp. 2273–2286, 1991.

[149] M. Mitrović and B. Tadić, “Spectral and dynamical properties in classes of sparse networks with mesoscopic inhomogeneities,” *Physical Review E*, vol. 80, p. 026123, 2009.

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

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

[152] M. E. J. Newman, “A measure of betweenness centrality based on random walks,” *Social Networks*, vol. 27, pp. 39–54, 2005.

[153] N. H. Nguyen, T. T. Do, and T. D. Tran, “A fast and efficient algorithm for low-rank approximation of a matrix,” in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pp. 215–224, 2009.

[154] J. Novembre, T. Johnson, K. Bryc, Z. Kutalik, A. R. Boyko, A. Auton, A. Indap, K. S. King, S. Bergmann, M. R. Nelson, M. Stephens, and C. D. Bustamante, “Genes mirror geography within Europe,” *Nature*, vol. 456, pp. 98–101, 2008.

[155] C. C. Paige and M. A. Saunders, “Algorithm 583: LSQR: Sparse linear equations and least-squares problems,” *ACM Transactions on Mathematical Software*, vol. 8, no. 2, pp. 195–209, 1982.

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

[157] C. T. Pan and P. T. P. Tang, “Bounds on singular values revealed by QR factorizations,” *BIT Numerical Mathematics*, vol. 39, pp. 740–756, 1999.

[158] F. Pan, X. Zhang, and W. Wang, “CRD: Fast co-clustering on large datasets utilizing sampling-based matrix decomposition,” in *Proceedings of the 34th SIGMOD international conference on Management of data*, pp. 173–184, 2008.

[159] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," *Journal of Computer and System Sciences*, vol. 61, no. 2, pp. 217–235, 2000.

[160] P. Parker, P. J. Wolfe, and V. Tarok, "A signal processing application of randomized low-rank approximations," in *Proceedings of the 13th IEEE Workshop on Statistical Signal Processing*, pp. 345–350, 2005.

[161] P. Paschou, P. Drineas, J. Lewis, C. M. N. D. A. Nickerson, J. D. Smith, P. M. Ridker, D. I. Chasman, R. M. Krauss, and E. Ziv, "Tracing sub-structure in the European American population with PCA-informative markers," *PLoS Genetics*, vol. 4, no. 7, p. e1000114, 2008.

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

[163] P. Paschou, M. W. Mahoney, A. Javed, J. R. Kidd, A. J. Pakstis, S. Gu, K. K. Kidd, and P. Drineas, "Intra- and interpopulation genotype reconstruction from tagging SNPs," *Genome Research*, vol. 17, no. 1, pp. 96–107, 2007.

[164] 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*, vol. 3, pp. 1672–1686, 2007.

[165] N. Patterson, A. L. Price, and D. Reich, "Population structure and eigenanalysis," *PLoS Genetics*, vol. 2, no. 12, pp. 2074–2093, 2006.

[166] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proceedings of the 8th Annual ACM SIGKDD Conference*, pp. 61–70, 2002.

[167] G. J. Rodgers and A. J. Bray, "Density of states of a sparse random matrix," *Physical Review B*, vol. 37, no. 7, pp. 3557–3562, 1988.

[168] V. Rokhlin, A. Szlam, and M. Tygert, "A randomized algorithm for principal component analysis," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1100–1124, 2009.

[169] V. Rokhlin and M. Tygert, "A fast randomized algorithm for overdetermined linear least-squares regression," *Proc. Natl. Acad. Sci. USA*, vol. 105, no. 36, pp. 13212–13217, 2008.

[170] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by local linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[171] M. Rudelson, "Random vectors in the isotropic position," *Journal of Functional Analysis*, vol. 164, no. 1, pp. 60–72, 1999.

[172] M. Rudelson and R. Vershynin, "Sampling from large matrices: An approach through geometric functional analysis," *Journal of the ACM*, vol. 54, no. 4, p. Article 21, 2007.

[173] Y. Saad, J. R. Chelikowsky, and S. M. Shontz, "Numerical methods for electronic structure calculations of materials," *SIAM Review*, vol. 52, no. 1, pp. 3–54, 2010.

[174] T. Sarlós, "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.

[175] L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee, "Spectral methods for dimensionality reduction," in *Semisupervised Learning*, (O. Chapelle, B. Schoelkopf, and A. Zien, eds.), pp. 293–308, MIT Press, 2006.

[176] B. Savas and I. Dhillon, "Clustered low rank approximation of graphs in information science applications," in *Proceedings of the 11th SIAM International Conference on Data Mining*, 2011.

[177] D. N. Spendley and P. J. Wolfe, "Adaptive beamforming using fast low-rank covariance matrix approximations," in *Proceedings of the IEEE Radar Conference*, pp. 1–5, 2008.

[178] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pp. 563–568, 2008.

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

[180] G. Strang, *Linear Algebra and Its Applications.* Harcourth Brace Jovanovich, 1988.

[181] 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.

[182] A. Talwalkar, S. Kumar, and H. Rowley, "Large-scale manifold learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

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

[184] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[185] The International HapMap Consortium, "The International HapMap Project," *Nature*, vol. 426, pp. 789–796, 2003.

[186] The International HapMap Consortium, "A haplotype map of the human genome," *Nature*, vol. 437, pp. 1299–1320, 2005.

[187] The Wellcome Trust Case Control Consortium, "Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls," *Nature*, vol. 447, pp. 661–678, 2007.

[188] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos, "Colibri: Fast mining of large static and dynamic graphs," in *Proceedings of the 14th Annual ACM SIGKDD Conference*, pp. 686–694, 2008.

[189] P. F. Velleman and R. E. Welsch, "Efficient computing of regression diagnostics," *The American Statistician*, vol. 35, no. 4, pp. 234–242, 1981.

[190] S. Venkatasubramanian and Q. Wang, "The Johnson-Lindenstrauss transform: An empirical study," in *ALENEX11: Workshop on Algorithms Engineering and Experimentation*, pp. 164–173, 2011.

[191] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular value decomposition and principal component analysis," in *A Practical Approach to Microarray*

Data Analysis, (D. P. Berrar, W. Dubitzky, and M. Granzow, eds.), pp. 91–109, Kluwer, 2003.

[192] 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*, pp. 682–688, 2001.

[193] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, "A fast randomized algorithm for the approximation of matrices," *Applied and Computational Harmonic Analysis*, vol. 25, no. 3, pp. 335–366, 2008.

[194] C. W. Yip, A. J. Connolly, A. S. Szalay, T. Budavári, M. SubbaRao, J. A. Frieman, R. C. Nichol, A. M. Hopkins, D. G. York, S. Okamura, J. Brinkmann, I. Csabai, A. R. Thakar, M. Fukugita, and Z. Ivezić, "Distributions of galaxy spectral types in the Sloan Digital Sky Survey," *The Astronomical Journal*, vol. 128, no. 2, pp. 585–609, 2004.

[195] C. W. Yip, A. J. Connolly, D. E. Vanden Berk, Z. Ma, J. A. Frieman, M. SubbaRao, A. S. Szalay, G. T. Richards, P. B. Hall, D. P. Schneider, A. M. Hopkins, J. Trump, and J. Brinkmann, "Spectral classification of quasars in the Sloan Digital Sky Survey: Eigenspectra, redshift, and luminosity effects," *The Astronomical Journal*, vol. 128, no. 6, pp. 2603–2630, 2004.

[196] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977.

[197] K. Zhang and J. T. Kwok, "Clustered Nyström method for large scale manifold learning and dimension reduction," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1576–1587, 2010.

[198] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved Nyström low-rank approximation and error analysis," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 1232–1239, 2008.