Preconditioned Data Sparsification for Big Data with Applications to PCA and K-means

Farhad Pourkamali-Anaraki and Stephen Becker

Abstract

We analyze a compression scheme for large data sets that randomly keeps a small percentage of the components of each data sample. The benefit is that the output is a sparse matrix and therefore subsequent processing, such as PCA or K-means, is significantly faster, especially in a distributed-data setting. Furthermore, the sampling is single-pass and applicable to streaming data. The sampling mechanism is a variant of previous methods proposed in the literature combined with a randomized preconditioning to smooth the data. We provide guarantees for PCA in terms of the covariance matrix, and guarantees for K-means in terms of the error in the center estimators at a given step. We present numerical evidence to show both that our bounds are nearly tight and that our algorithms provide a real benefit when applied to standard test data sets, as well as providing certain benefits over related sampling approaches.

Index Terms

Unsupervised learning, PCA, clustering, K-means, randomized algorithm, big data.

I. INTRODUCTION

Principal component analysis (PCA) is a classical unsupervised analysis method commonly used in all quantitative disciplines [1]. Given n observations of p-dimensional data $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^p$, standard algorithms to compute PCA require $\mathcal{O}(p^2 n)$ flops (we assume throughout the paper that $n \geq p$), which is expensive for large p and/or n. The problem is exacerbated in situations with high communication cost, as in distributed data settings such as Hadoop/MapReduce clusters or as in sensor networks. An extreme case is when the data rate is so large that we turn to "streaming" algorithms which examine each new data entry and then discard it, i.e., take at most one pass over the data [2].

Another commonly used unsupervised analytic task is clustering, which refers to identifying groups of similar data samples in a data set. Running a simple clustering method such as the K-means algorithm turns out to be infeasible for distributed and streaming data [3]. Therefore, a recent line of research aims at developing scalable learning tools for "big data" and providing fundamental insights and tradeoffs involved in these algorithms.

This paper proposes a specific mechanism to reduce the computational, storage, and communication burden in unsupervised analysis methods such as PCA and K-means clustering, which requires only one pass over the data. The first step is a pre-processing step designed to precondition the data and smooth out large entries, which is based on the well-known fast Johnson-Lindenstrauss result [4]. We summarize existing results about this preconditioning and describe how it benefits our approach. The second step is element-wise random sampling: choosing to keep exactly m out of p entries (without replacement) per sample. This step can be viewed as forming a sampling matrix $\mathbf{R}_i \in \mathbb{R}^{p \times m}$ which contains m distinct canonical basis vectors drawn uniformly at random, and multiplying each preconditioned sample $\mathbf{x}_i$ on the left by $\mathbf{R}_i\mathbf{R}_i^T$ (the projection matrix onto the subspace spanned by the columns of $\mathbf{R}_i$). The two steps of our approach, preconditioning and sampling, can be easily combined into single pass over the data which eliminates the need to revisit past entries of the data.

We then analyze, in a non-Bayesian setting, the performance of our estimators and provide theorems that show as n grows with p fixed, the bounds hold with high probability [1] if $m = \mathcal{O}(\log n / n)$, cf. Corollary 5. This means that as we collect more samples, the size of the incoming data increases proportional to n but the amount of data we must store only increases like $\log n$, i.e., we compress with ratio $n / \log n$. This is possible because of the careful sampling scheme — if one were to simply keep some of the data points $\mathbf{x}_i$, then for a small error in the ℓ_∞ norm, one needs to keep a *constant* fraction of the data.

Similar proposals, based on dimensionality reduction or sampling, have been proposed for analyzing large data sets, e.g., [6]–[8]. Many of these schemes are based on multiplying the data on the left by a single random matrix $\mathbf{\Omega}$ and recording $\mathbf{\Omega}\mathbf{X}$ where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ is the $p \times n$ matrix with data samples as columns, $\mathbf{\Omega}$ is $m \times p$ with $m < p$, and the columns of $\mathbf{\Omega}\mathbf{X}$ are known as sketches or compressive measurements. The motivation behind sketching is the classical Johnson-Lindenstrauss lemma [9] which states that the scaled pairwise distances between low-dimensional sketches are preserved to within a small tolerance for some random matrices $\mathbf{\Omega}$. Moreover, the more recent field of compressive sensing demonstrates that low-dimensional

Farhad Pourkamali-Anaraki is with the Department of Electrical, Computer, and Energy Engineering, University of Colorado Boulder, Boulder, CO 80309 USA (e-mail: farhad.pourkamali@colorado.edu).

Stephen Becker is with the Department of Applied Mathematics, University of Colorado Boulder, Boulder, CO 80309 USA (e-mail: stephen.becker@colorado.edu).

[1] in the sense of [5]

compressive measurements $\mathbf{\Omega X}$ contain enough information to recover the data samples under the assumption that $\mathbf{X}$ is sparse in some known basis representation [10].

However, for PCA, the desired output is the eigendecomposition of $\mathbf{XX}^T$ or, equivalently, the left singular vectors of $\mathbf{X}$, for which we cannot typically impose structural constraints. Clearly, a single random matrix $\mathbf{\Omega}^T \in \mathbb{R}^{p \times m}$ with $m < p$ only spans at most an m-dimensional subspace of $\mathbb{R}^p$, and after the transformation $\mathbf{\Omega X}$, information about the components of the left singular vectors within the orthogonal complement subspace will be lost. Another technique to recover the left singular vectors is to also record $\mathbf{X\Omega}'$ for another random matrix $\mathbf{\Omega}'$ of size $n \times m'$ with $m' < n$ [7], and with careful implementation, it is possible to record $\mathbf{\Omega X}$ and $\mathbf{X\Omega}'$ in a one-pass algorithm. However, even in this case, it is not possible to return something as simple as a center estimator in the K-means algorithm without making a second pass over the data.

Our approach circumvents this since each data sample $\mathbf{x}_i$ is multiplied by its own random matrix $\mathbf{R}_i\mathbf{R}_i^T$, where $\mathbf{R}_i \in \mathbb{R}^{p \times m}$ consists of a randomly chosen subset of m distinct canonical basis vectors. We show that our scheme leads to one-pass algorithms for unsupervised analysis methods such as PCA and K-means clustering, meanwhile containing enough information to recover principal components and cluster centers without imposing additional structural constraints on them.

We expand on comparisons with relevant literature in more detail in Section II, summarize important results about the preconditioning transformation and sub-sampling in Section III, provide theoretical results on the sample mean estimator and covariance estimator in sections IV and V respectively, then focus on results relevant for K-means clustering in Section VI and finish the paper with numerical experiments in Section VII and a conclusion.

A. Setup

In this paper, we consider a non-Bayesian data setting where we make no distributional assumptions on the set of data samples $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$. For each sample $\mathbf{x}_i$, we form a sampling matrix $\mathbf{R}_i \in \mathbb{R}^{p \times m}$, where the m columns are chosen uniformly at random from the set of canonical basis vectors in $\mathbb{R}^p$ without replacement. Thus, we use m for the intrinsic dimensionality of the compressed samples, and write the compression factor as $\gamma = \frac{m}{p}$.

We name column vectors by lower-case bold letters and matrices by upper-case bold letters. Our results will involve bounds on the Euclidean and maximum norm in $\mathbb{R}^p$, denoted $\|\mathbf{x}\|_2$ and $\|\mathbf{x}\|_\infty$ respectively, as well as the spectral and Frobenius norms on the space of $p \times n$ matrices, denoted $\|\mathbf{X}\|_2$ and $\|\mathbf{X}\|_F$ respectively. Recall that $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{p}\|\mathbf{x}\|_\infty$. We use $\|\mathbf{X}\|_{\max}$ to denote the maximum absolute value of the entries of a matrix, $\|\mathbf{X}\|_{\text{max-row}}$ to be the maximum ℓ_2 norm of the rows of a matrix, i.e., $\|\mathbf{X}\|_{\text{max-row}} = \|\mathbf{X}\|_{2 \to \infty} = \sup_{\mathbf{y} \neq 0} \frac{\|\mathbf{Xy}\|_\infty}{\|\mathbf{y}\|_2}$, and $\|\mathbf{X}\|_{\text{max-col}} = \|\mathbf{X}\|_{1 \to 2}$ to be the maximum ℓ_2 norm of the columns of a matrix.

Let $\mathbf{e}_i$ denote the i-th vector of the canonical basis in $\mathbb{R}^p$, where entries are all zero except for the i-th one which is 1. In addition, $\text{diag}(\mathbf{x})$ returns a square diagonal matrix with the entries of vector $\mathbf{x}$ on the main diagonal, and $\text{diag}(\mathbf{X})$ denotes the matrix formed by zeroing all but the diagonal entries of matrix $\mathbf{X}$. We also represent the entry in the i-th row and the j-th column of matrix $\mathbf{X}$ as $X_{i,j}$.

B. Contributions

In this paper, we introduce an efficient data sparsification framework for large-scale data applications that does not require incoherence and distributional assumptions on the data. Our approach exploits randomized orthonormal systems to find informative low-dimensional representations in a single pass over the data. Thus, our proposed compression technique can be used in streaming applications, where data samples cannot fit into memory [11]. We present results on the properties of sampling matrices and randomized orthonormal systems in Theorem B4 and Section III, respectively.

To demonstrate the effectiveness of our framework, we investigate the application of preconditioned data sparsification in two important unsupervised analysis methods, PCA and K-means clustering. Two unbiased estimators are introduced to recover the sample mean and covariance matrix from the compressed data. We provide strong theoretical guarantees on the closeness of the estimated and true sample mean in Theorem 4. Moreover, we employ recent results on exponential concentration inequalities for matrices [12] to obtain an exponentially decaying bound on the probability that the estimated covariance matrix deviates from its mean value in Theorem 6. Numerical simulations are provided to validate the tightness of the concentration bounds derived in this paper. Furthermore, our theoretical results reveal the connections between accuracy and important parameters, e.g., the compression factor and the underlying structure of data.

We also examine the application of our proposed data sparsification technique in the K-means clustering problem. In this case, we first define an optimal objective function based on the Maximum-Likelihood estimation. Then, an algorithm called sparsified K-means is introduced to find assignments as well as cluster centers in one pass over the data. Theoretical guarantees on the clustering structure per iteration are provided in Theorem 8. Finally, we present extensive numerical experiments to demonstrate the effectiveness of our sparsified K-means algorithm on large-scale data sets containing up to 10 million samples.

Beyond just the two unsupervised learning examples, we advocate the general usage of preconditioning followed by sampling. Sampling methods have long been popular due to their simplicity, but unless appropriate weighted sampling distributions are used, the accuracy of these methods is low. The use of the preconditioning obviates the need for weighted distributions, and is easy to analyze and implement.

II. Related Work

Our work has a close connection to the recent line of research on signal processing and information retrieval from compressive measurements that assumes one only has access to low-dimensional random projections of the data [13]–[15]. For example, Eldar and Gleichman [16] studied the problem of learning basis representations (dictionary learning) for the data samples $\mathbf{X} \in \mathbb{R}^{p \times n}$ under the assumption that a single random matrix $\mathbf{R} \in \mathbb{R}^{p \times m}$, $m < p$, is used for all the samples. It was shown that $\mathbf{R}^T \mathbf{X}$ does not contain enough information to recover the basis representation for the original data samples, unless structural constraints such as sparsity over the set of admissible representations are imposed. This mainly follows from the fact that $\mathbf{R}$ has a non-trivial null space and, without imposing constraints, we cannot recover the information about the whole space $\mathbb{R}^p$. For example, consider a simple case of a rank-one data matrix, $\mathbf{X} = \sigma \mathbf{u} \mathbf{v}^T$, where the goal is to recover the single principal component $\mathbf{u} \in \mathbb{R}^p$ from $\mathbf{R}^T \mathbf{X}$. The singular value decomposition of $\mathbf{R}^T \mathbf{X}$ results in $\mathbf{R}^T \mathbf{X} = \sigma \mathbf{u}' \mathbf{v}^T$, where $\mathbf{u}' = \mathbf{R}^T \mathbf{u} \in \mathbb{R}^m$, so we have retained information on $\mathbf{v}$. However, accurate estimation of $\mathbf{u}$ from $\mathbf{u}' = \mathbf{R}^T \mathbf{u}$ is not possible, unless additional constraints are imposed on $\mathbf{u}$. The line of work [17]–[20] addressed this issue, in the dictionary learning setting, by observing each data sample $\mathbf{x}_i$ through *distinct* random matrices.

Another line of work considers covariance estimation and recovery of principal components (PCs) based on compressive measurements [21]–[26]. In particular, Qi and Hughes [22] proposed a method for recovering the mean and principal components of $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ from compressive measurements $\mathbf{R}_i^T \mathbf{x}_i \in \mathbb{R}^m$, $i = 1, \dots, n$, where $\mathbf{R}_i \in \mathbb{R}^{p \times m}$ with $m < p$ is a random matrix with entries drawn i.i.d. from the Gaussian distribution. This method requires computing the projection matrix onto the subspace spanned by the columns of $\mathbf{R}_i$ which is $\mathbf{P}_i = \mathbf{R}_i (\mathbf{R}_i^T \mathbf{R}_i)^{-1} \mathbf{R}_i^T$. Then, each $\mathbf{P}_i \mathbf{x}_i$ can be directly computed from the compressive measurements since $\mathbf{P}_i \mathbf{x}_i = \mathbf{R}_i (\mathbf{R}_i^T \mathbf{R}_i)^{-1} \mathbf{R}_i^T \mathbf{x}_i$. It has been shown that the mean and principal components of $\mathbf{P}_i \mathbf{x}_i \in \mathbb{R}^p$, $i = 1, \dots, n$, converge to the mean and principal components of the original data (up to a known scaling factor) as the number of data samples n goes to the infinity. However, this method is computationally inefficient because even if each (pseudo-)random matrix $\mathbf{R}_i$, $i = 1, \dots, n$, is implicitly stored by a seed, the computational cost of the matrix multiplies and inversions is high.

Their work was extended in [25], where the authors studied the problem of performing PCA on $\mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i$ (eliminating the matrix inversion) for a general class of random matrices, where the entries of $\mathbf{R}_i$ are drawn i.i.d. from a zero-mean distribution with finite moments. Two estimators were proposed to estimate the mean and principal components of the original data with statistical guarantees for a finite number of samples n. Moreover, it was shown that one can use very sparse random matrices [27] to increase the efficiency in large-scale data sets. However, generating n unstructured random matrices, with $p \times m$ entries in each, can still be memory/computation inefficient for high-dimensional data sets with p large. Another disadvantage of the prior work [22] and [25] is that the result only holds for data samples drawn from a specific probabilistic generative model known as the spiked covariance model, which may not be informative for real-world data sets.

In the fields of theoretical computer science and numerical linear algebra, there are several alternative lines of work that are related to our proposed approach and we discuss them below in detail.

A. Comparison with column sampling approaches

The most natural compression scheme for large-scale data sets would be to directly select a small subset of the original data and then perform data analytic tasks on the reduced subset. The two natural distributions for column sampling are the uniform distribution and a non-uniform data-dependent distribution based on so-called statistical leverage scores [8][2]. The former method, uniform sampling, is a simple one-pass algorithm but it will perform poorly on many problems if there is any structural non-uniformity in the data [28].

To see this, we recreate the numerical experiment of [28] and compare the accuracy of left singular vectors/principal components (PCs) estimated using our proposed approach with those estimated after uniform column sampling. We set the parameters $p = 512$ and $n = 1024$ and consider 1000 runs for different values of the compression factor $\gamma = \frac{m}{p}$. In each run, we generate a data matrix $\mathbf{X} \in \mathbb{R}^{p \times n}$ from the multivariate t-distribution with 1 degree of freedom and covariance matrix $\mathbf{C}$ where $C_{ij} = 2 \times 0.5^{|i-j|}$. In our approach, we precondition the data as described in §III and then keep exactly m out of p entries for each data sample to obtain a sparse matrix. To have a fair comparison, we consider randomly selecting $2m$ columns of $\mathbf{X}$ because $n/p = 2$ and our sparse matrix has exactly $2mp$ nonzero entries. We measure the accuracy of the estimated PCs based on the explained variance [11]: given estimates of k PCs $\widehat{\mathbf{U}} \in \mathbb{R}^{p \times k}$ (we take $k = 10$), the fraction of explained variance is defined as $\mathrm{tr}(\widehat{\mathbf{U}}^T \mathbf{X} \mathbf{X}^T \widehat{\mathbf{U}}) / \mathrm{tr}(\mathbf{X} \mathbf{X}^T)$, and closeness of this fraction to 1 represents higher accuracy.

Fig. 1 reports the mean and standard deviation of the explained variance over 1000 trials. For both approaches, the average explained variance approaches 1 as $\gamma \to 1$, as expected, and uniform column sampling is slightly more accurate than our approach. However, uniform column sampling has an extremely high variance for all values of γ, e.g., the standard deviations for $\gamma = 0.1$, 0.2, 0.3 are 0.20, 0.28, and 0.31 respectively. On the other hand, the standard deviation for our approach is significantly smaller for all values of γ (less than 0.04), due to the preconditioning. Thus the *worst-case* performance of our approach is reasonable, while the worst-case performance of column sampling may be catastrophically bad.

---

[2]Sampling according to column norm is a common third option, but is generally inferior to leverage-score based sampling
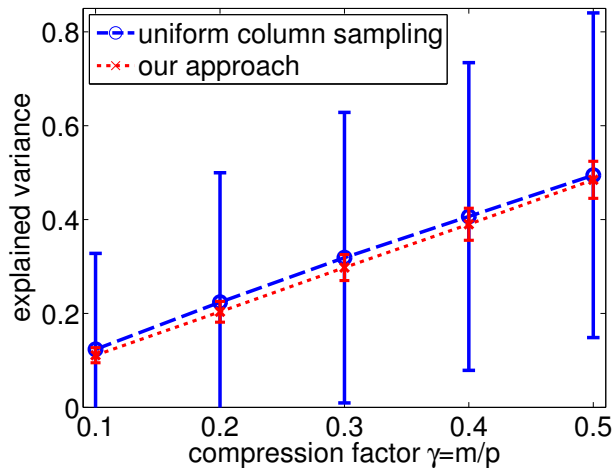
Figure 1. Accuracy of estimated PCs via one-pass methods: uniform column sampling and our proposed precondition+sparsification approach. We plot the mean and standard deviation of the explained variance over 1000 runs for each value of γ. The standard deviation for our approach is significantly smaller compared to the uniform column sampling.

The second common method for column sampling is based on a data-dependent non-uniform distribution and has received great attention in the development of randomized algorithms such as low-rank matrix approximation. These data-dependent sampling techniques are typically computationally expensive and a SVD on the data matrix is required. Recently, there are variants that can be computed more efficiently such as [29]. However, these algorithms compute the sampling distribution in at least one pass of all the samples, and a second pass is required to actually sample the columns based on this distribution. Thus at least two passes are required and these algorithms are not suitable for streaming [30].

B. Comparison with other element-wise sampling approaches

Analysis of sparsification of matrices for the purpose of fast computation of low-rank approximations goes back to at least Achlioptas and McSherry [31], [32], who propose independently keeping each entry of the matrix in either a uniform fashion or a non-uniform fashion. In the former, each entry of the data matrix is kept with the same probability, whereas in the non-uniform case, each entry is kept with probability proportional to its magnitude squared. Under the latter scheme, the expected number of nonzero entries can be bounded but one does not have precise control on the exact number of nonzero entries.

Recently, Achlioptas et al. [33] have considered a variant of element-wise sampling where a weight is assigned to each entry of the data matrix according to its absolute value normalized by the ℓ_1 norm of the corresponding row, and then a fixed number of entries is sampled (with replacement) from the matrix. Computing the exact ℓ_1 norms of the rows requires one pass over the data and consequently this method requires two passes over the data. It is shown empirically that disregarding the normalization factor performs quite well in practice, thus yielding a one-pass algorithm. Calculating guarantees on their one-pass variant is an interesting open-problem.

C. K-means clustering for big data

Clustering is a commonly used unsupervised learning task that reveals the underlying structure of a data set by splitting the data into groups, or clusters, of similar samples. It has applications ranging from search engines and social network analysis to medical and e-commerce domains. Among clustering algorithms, K-means [34] is one of the most popular clustering algorithms [35]. K-means is an iterative expectation-maximization type algorithm in which each cluster is associated with a representative vector or cluster center, which is known to be the sample mean of the vectors in that cluster. In each iteration of K-means, data samples are assigned to the nearest cluster centers (usually based on the Euclidean norm) and cluster centers are then updated based on the most recent assignment of the data. Therefore, the goal of K-means is to find a set of cluster centers as well as assignment of the data.

Despite the simplicity of K-means, there are several challenges in dealing with modern large-scale data sets. The high dimensionality and large volumes of data make it infeasible to store the full data in a centralized location or communicate the data in distributed settings [3] and these voluminous data sets make K-means computationally inefficient [36].

Recent approaches to K-means clustering for big data have focused on selecting the most informative features of the data set and performing K-means on the reduced set. For example, Feldman et al. [37] introduced coresets for K-means, in which one reduces the dimension of the data set from p to m by projecting the data on the top m left singular vectors (principal components) of $\mathbf{X}$, although this requires the SVD of the data matrix and so does not easily apply to streaming scenarios. To reduce the cost of computing exact SVD, the authors in [38] proposed using approximate SVD algorithms instead.

Other prior works, such as [36], [39], have used randomized schemes for designing efficient clustering algorithms. In particular, Mahoney et al. [36] proposed two provably accurate algorithms to reduce the dimension of the data by selecting a small subset of m rows of the data matrix $\mathbf{X} \in \mathbb{R}^{p \times n}$ (feature selection) or multiplying the data matrix from the left by a random matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times p}$ (feature extraction). Afterwards, the K-means algorithm is applied on these n data samples in $\mathbb{R}^m$ to find the assignment of the original data. In feature selection, the rows of $\mathbf{X}$ are sampled via a non-uniform distribution, which requires two passes over the data. Then, having the distribution, it requires one more pass to actually sample m rows which leads to a three-pass algorithm.

Furthermore, in both feature selection and feature extraction algorithms, K-means is applied on the projected data in $\mathbb{R}^m$ so there is no ready estimate of the cluster centers in the original space $\mathbb{R}^p$. One could transform the cluster centers to $\mathbb{R}^p$ with the pseudo-inverse of $\mathbf{\Omega}$ (recall $\mathbf{\Omega}$ is low-rank) but this estimate is poor; see Figs. 9g and 9i in §VII. A better approach to calculate the cluster centers is to use the calculated assignment of vectors in the original space, which requires one additional pass over the data, and consequently neither feature selection nor feature extraction is streaming.

In this paper, we address the need to account for the computational/storage burden and we propose a randomized algorithm for K-means clustering, called sparsified K-means, that returns both the cluster centers and the assignment of the data in single pass over the data. To do this, we take a different approach where a randomized unitary transformation is first applied to the data matrix and we then choose m out of p entries of each preconditioned data uniformly at random to obtain a sparse matrix. Afterwards, the K-means algorithm is applied on the resulting sparse matrix and, consequently, speedup in processing time and savings in memory are achieved based on the value of compression factor $\gamma = m/p < 1$.

We provide theoretical guarantees on the clustering structure of our sparsified K-means algorithm compared to K-means on the original data in each iteration. In fact, this is another advantage of our method over the previous work [36], where guarantees are available only for the final value of the *objective function* and, thus, it is not possible to directly compare the clustering structure of feature selection and feature extraction algorithms with K-means on the original data.

III. PRELIMINARIES

The randomized orthonormal system (ROS) is a powerful randomization tool that has found uses in fields from machine learning [4] to numerical linear algebra [7] and compressive sensing [40]. We use the ROS to efficiently precondition the data and smooth out large entries in the matrix $\mathbf{X}$ before sampling. It is straightforward to combine this preconditioning and sampling operation into a single pass on the data. Furthermore, we can unmix the preconditioned data because the preconditioning transformation is unitary, so by applying its adjoint we undo its effect. Because the operator is unitary, it does not affect our estimates in the Euclidean norm (for vectors) or the spectral and Frobenius norms (for matrices). Moreover, this transformation is stored implicitly and based off fast transforms, so applying to a length p vector takes $\mathcal{O}(p \log(p))$ complexity, and applying it to a matrix is embarrassingly parallel across the columns [41], [42].

Specifically, the ROS preconditioning transformation uses matrices $\mathbf{H}$ and $\mathbf{D}$ and is defined

$$\mathbf{x} \mapsto \mathbf{y} = \mathbf{HDx} \tag{1}$$

where $\mathbf{H} \in \mathbb{R}^{p \times p}$ is a deterministic orthonormal matrix such as a Hadamard, Fourier, or DCT matrix for which matrix-vector multiplication can be implemented efficiently in $\mathcal{O}(p \log(p))$ complexity without the need to store the matrices explicitly. The matrix $\mathbf{D} \in \mathbb{R}^{p \times p}$ is a stochastic diagonal matrix whose entries on the main diagonal are random variables drawn uniformly from $\{\pm 1\}$. The matrix product $\mathbf{HD} \in \mathbb{R}^{p \times p}$ is an orthonormal matrix and this mapping ensures that, with high probability, the magnitude of any entry of $\mathbf{HDx}$ is about $\mathcal{O}(1/\sqrt{p})$ for any unit vector $\mathbf{x}$ (cf. Thm. 1).

We now present relevant results about the preconditioning transformation $\mathbf{HD}$ (1) that will be used in the next sections; more sophisticated results are in [43] and Appendix D.

Theorem 1 (Single element bound for ROS (1)). *Let* $\mathbf{x} \in \mathbb{R}^p$, *and* $\mathbf{y} = \mathbf{HDx}$ *a random variable from the ROS applied to* $\mathbf{x}$, *then for every* $j = 1, \ldots, p$,

$$\mathbb{P}\left\{|y_j| \geq (t/\sqrt{p})\|\mathbf{x}\|_2\right\} \leq 2 \exp\left(-\eta t^2/2\right) \tag{2}$$

where $\eta = 1$ *for* $\mathbf{H}$ *a Hadamard matrix and* $\eta = 1/2$ *for* $\mathbf{H}$ *a DCT matrix.*

As pointed out in [43], the proof follows easily from Hoeffding's inequality (Thm. A1). Since $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2$, the theorem implies that no single entry y_j is likely to be far away from the average $\|\mathbf{y}\|_2/\sqrt{p}$.

Using the union bound, we derive the following results:

Corollary 2. *Let* $\mathbf{X}$ *be a* $p \times n$ *matrix with normalized columns, and* $\mathbf{Y} = \mathbf{HDX}$ *a random variable from the ROS applied to* $\mathbf{X}$, *then for all* $\alpha \in (0, 1)$,

$$\mathbb{P}\left\{\|\mathbf{Y}\|_{max} \geq \frac{1}{\sqrt{p}} \cdot \sqrt{\frac{2}{\eta} \log\left(\frac{2np}{\alpha}\right)}\right\} \leq \alpha \tag{3}$$

$$\mathbb{P}\left\{\|\mathbf{Y}\|_{max\text{-}row} \geq \sqrt{\frac{n}{p}} \cdot \sqrt{\frac{2}{\eta} \log\left(\frac{2np}{\alpha}\right)}\right\} \leq \alpha \tag{4}$$

Proof: Taking the union bound over all np entries in the matrix $\mathbf{Y}$ gives (for each column of $\mathbf{X}$, we have $\|\mathbf{x}_i\|_2 = 1$)

$$\mathbb{P}\left\{\|\mathbf{Y}\|_{\max} \geq t/\sqrt{p}\right\} \leq 2np \exp\left(-\eta t^2/2\right)$$

which leads to (3) if we choose $t^2 = \frac{2}{\eta}\log(2np/\alpha)$. To derive the second equation, let $\mathbf{Y}_{j,:}$ denote the j-th row of $\mathbf{Y}$. We apply the union bound to get

$$\mathbb{P}\left\{\|\mathbf{Y}_{j,:}\|_\infty \geq t/\sqrt{p}\right\} \leq 2n \exp\left(-\eta t^2/2\right)$$

and we bound the ℓ_2 norm of $\mathbf{Y}_{j,:} \in \mathbb{R}^{1\times n}$ with $\sqrt{n}$ times the ℓ_∞ norm,

$$\mathbb{P}\left\{\|\mathbf{Y}_{j,:}\|_2 \geq \sqrt{\frac{n}{p}} \cdot t\right\} \leq 2n \exp\left(-\eta t^2/2\right) \tag{5}$$

from which (4) follows by taking the union bound over p rows and again choosing $t^2 = \frac{2}{\eta}\log(2np/\alpha)$. $\blacksquare$

Note that for $\mathbf{H}$ a Hadamard matrix, the result in (5) shows the square of the ℓ_2 norm of $\mathbf{Y}_{j,:}$ is unlikely to be larger than its mean value. To see this, note that the (u,l) entry of the Hadamard matrix $H_{u,l}$ is either $\frac{1}{\sqrt{p}}$ or $-\frac{1}{\sqrt{p}}$, and $\mathbf{D} = \text{diag}([D_1, \ldots, D_p])$ where each D_i is ± 1 with equal probability. Without loss of generality, we consider the first row of $\mathbf{Y}$, $\mathbf{Y}_{1,:}$, and find the expectation of its k-th element squared:

$$\mathbb{E}[Y_{1,k}^2] = \mathbb{E}\left[\left(\sum_{l=1}^p D_l H_{1,l} X_{l,k}\right)^2\right]$$
$$= \sum_{l=1}^p \mathbb{E}[D_l^2] H_{1,l}^2 X_{l,k}^2$$
$$+ \sum_{l_1 \neq l_2} \mathbb{E}[D_{l_1} D_{l_2}] H_{1,l_1} H_{1,l_2} X_{l_1,k} X_{l_2,k}$$
$$= \frac{1}{p}$$

since $\mathbb{E}[D_l] = 0$, $\mathbb{E}[D_l^2] = 1$, and $\mathbf{X}$ has normalized columns, i.e., $\sum_{l=1}^p X_{l,k}^2 = 1$. Thus, we get $\mathbb{E}[\|\mathbf{Y}_{1,:}\|_2^2] = \sum_{k=1}^n \mathbb{E}[Y_{1,k}^2] = \frac{n}{p}$.

The importance of the preconditioning by the ROS prior to sub-sampling is the reduction of the norms over the worst-case values. As we will see in the subsequent sections, the variance of our sample mean and covariance estimators depends on the quantities such as the maximum absolute value of the entries and the maximum ℓ_2 norm of the rows of the data matrix. Therefore, the preconditioning step (1) is essential to obtain accurate and reliable estimates. For example, let us consider a data matrix $\mathbf{X}$ with normalized columns, then it is possible for $\|\mathbf{X}\|_{\max} = 1$, which would lead to weak bounds when inserted into our estimates. The best possible norm is $1/\sqrt{p}$ which occurs if all entries have the same magnitude. Our result in Corollary 2 states that, under the ROS, with high probability all the entries of $\mathbf{Y} = \mathbf{HDX}$ have comparable magnitude and it is unlikely $\|\mathbf{Y}\|_{\max}$ is larger than $\sqrt{\log(np)}/\sqrt{p}$, which leads to lower variance and improved accuracy of our estimates.

Next, we present a result about the effect of sub-sampling on reduction of the Euclidean norm. Let $\mathbf{w} = \mathbf{RR}^T\mathbf{x}$ denote the sub-sampled version of $\mathbf{x} \in \mathbb{R}^p$. Obviously, $\|\mathbf{w}\|_2^2 \leq \|\mathbf{x}\|_2^2$ and for data sets with a few large entries, $\|\mathbf{w}\|_2^2$ may be very close to $\|\mathbf{x}\|_2^2$ meaning that sub-sampling has not appreciably decreased the Euclidean norm. For example, consider a vector $\mathbf{x} = [1, 0.1, 0.01, 0.001]^T$ where we wish to sample two out of four entries uniformly at random. In this case, $\|\mathbf{w}\|_2^2$ takes extreme values such that it might be very close to either $\|\mathbf{x}\|_2^2$ or zero. However, if we precondition the vector $\mathbf{x}$ prior to sub-sampling based on (1), this almost never happens, and the norm is reduced by nearly m/p as one would hope for:

Corollary 3 (Norm bounds for ROS (1)). *Let $\mathbf{x} \in \mathbb{R}^p$, and $\mathbf{y} = \mathbf{HDx}$ a random variable from the ROS applied to $\mathbf{x}$. Define $\mathbf{w} \in \mathbb{R}^p$ to be a sampled version of $\mathbf{y}$, keeping m of p entries uniformly at random (without replacement). Then with probability greater than $1 - \alpha$,*

$$\|\mathbf{w}\|_2^2 \leq \frac{m}{p}\frac{2}{\eta}\log\left(\frac{2p}{\alpha}\right)\|\mathbf{x}\|_2^2 \tag{6}$$

where $\eta = 1$ for $\mathbf{H}$ a Hadamard matrix and $\eta = 1/2$ for $\mathbf{H}$ a DCT matrix. Moreover, if $\{\mathbf{w}_i\}_{i=1}^n \in \mathbb{R}^p$ are sampled versions of the preconditioned data $\mathbf{Y} = \mathbf{HDX}$, with probability greater than $1 - \alpha$,

$$\|\mathbf{w}_i\|_2^2 \leq \frac{m}{p}\frac{2}{\eta}\log\left(\frac{2np}{\alpha}\right)\|\mathbf{x}_i\|_2^2, \ i = 1, \ldots, n. \tag{7}$$

Proof: Regardless of how we sample, it holds deterministically that $\|\mathbf{w}\|_2 \leq \sqrt{m}\|\mathbf{y}\|_\infty$ and this bound is reasonably sharp since the entries of $\mathbf{y}$ are designed to have approximately the same magnitude. Using Thm. 1 and the union bound, the probability that $\|\mathbf{y}\|_\infty \geq (t/\sqrt{p})\|\mathbf{x}\|_2$ is less than $2pe^{-\eta t^2/2}$. Then, we choose $t^2 = \frac{2}{\eta}\log(2p/\alpha)$. Finally, we use the union bound when we have n data samples. $\blacksquare$

As a result of Corollary 3, when original data samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are first preconditioned and then sub-sampled, by choosing $\alpha = 1/100$ we see that $\|\mathbf{w}_i\|_2^2 \leq \frac{m}{p}\frac{2}{\eta}\log(200np)\|\mathbf{x}_i\|_2^2$, $i = 1, \ldots, n$, with probability greater than 0.99.

IV. THE SAMPLE MEAN ESTIMATOR

We show that a rescaled version of the sample mean of $\{\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\}_{i=1}^n$, where m out of p entries of $\mathbf{x}_i$ are kept uniformly at random without replacement, is an unbiased estimator for the sample mean of the full data $\{\mathbf{x}_i\}_{i=1}^n$. We will upper bound the error in both ℓ_∞ and ℓ_2 norms, and show that these bounds are worse when the data set has a few large entries, which motivates our preconditioning.

Theorem 4. *Let $\overline{\mathbf{x}}_n$ represent the sample mean of $\{\mathbf{x}_i\}_{i=1}^n$, i.e., $\overline{\mathbf{x}}_n = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$. Construct a rescaled version of the sample mean from $\{\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\}_{i=1}^n$, where each column of $\mathbf{R}_i \in \mathbb{R}^{p\times m}$ is chosen uniformly at random from the set of all canonical basis vectors without replacement:*

$$\widehat{\overline{\mathbf{x}}}_n = \frac{p}{m}\frac{1}{n}\sum_{i=1}^n \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i. \tag{8}$$

Then, $\widehat{\overline{\mathbf{x}}}_n$ is an unbiased estimator for $\overline{\mathbf{x}}_n$, i.e., $\mathbb{E}[\widehat{\overline{\mathbf{x}}}_n] = \overline{\mathbf{x}}_n$. Moreover, defining $\tau(m,p)$ as follows:

$$\tau(m,p) := \max\left\{\left(\frac{p}{m}-1\right), 1\right\} = \begin{cases} \left(\frac{p}{m}-1\right) & \text{if } \frac{m}{p} \le 0.5 \\ 1 & \text{if } \frac{m}{p} > 0.5 \end{cases} \tag{9}$$

then for all $t \ge 0$, with probability at least $1-\delta_1$,

$$\delta_1 = 2p\exp\left(\frac{-nt^2/2}{(\frac{p}{m}-1)\|\mathbf{X}\|_{\text{max-row}}^2/n + \tau(m,p)\|\mathbf{X}\|_{\text{max}}t/3}\right) \tag{10}$$

we have the following ℓ_∞ result:

$$\|\widehat{\overline{\mathbf{x}}}_n - \overline{\mathbf{x}}_n\|_\infty \le t. \tag{11}$$

Proof: First, we show that $\widehat{\overline{\mathbf{x}}}_n$ is an unbiased estimator:

$$\mathbb{E}[\widehat{\overline{\mathbf{x}}}_n] = \frac{p}{m}\frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[\mathbf{R}_i\mathbf{R}_i^T\right]\mathbf{x}_i = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i = \overline{\mathbf{x}}_n$$

where we used Thm. B4. To upper bound the error, we define:

$$\mathbf{u} := \widehat{\overline{\mathbf{x}}}_n - \overline{\mathbf{x}}_n = \sum_{i=1}^n \frac{1}{n}\left(\frac{p}{m}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \mathbf{x}_i\right) \tag{12}$$

and, thus, the j-th entry of $\mathbf{u} = [u_1,\ldots,u_p]^T$ can be written as a sum of independent centered random variables:

$$u_j = \sum_{i=1}^n z_i, \text{ where } z_i = \frac{1}{n}\mathbf{e}_j^T\left(\frac{p}{m}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \mathbf{x}_i\right). \tag{13}$$

We now use the Bernstein inequality (Thm. A2) to show each entry of $\mathbf{u}$ is concentrated around zero with high probability. To do this, let us define $\tau(m,p) := \max\{(\frac{p}{m}-1), 1\}$. We observe that each z_i is bounded:

$$|z_i| \le \frac{1}{n}\left\|\frac{p}{m}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \mathbf{x}_i\right\|_\infty \le \frac{1}{n}\tau(m,p)\|\mathbf{X}\|_{\text{max}} \tag{14}$$

since $\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i$ is a vector with m entries of $\mathbf{x}_i$ and the rest equal to zero. Next, we find the variance of z_i, $\mathbb{E}[z_i^2] = (1/n^2)\mathbf{e}_j^T\mathbf{\Lambda}\mathbf{e}_j$, where we use Thm. B4 to compute $\mathbf{\Lambda}$:

$$\begin{aligned}
\mathbf{\Lambda} &= \mathbb{E}\left[\left(\frac{p}{m}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \mathbf{x}_i\right)\left(\frac{p}{m}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \mathbf{x}_i\right)^T\right] \\
&= \frac{p^2}{m^2}\mathbb{E}\left[\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{R}_i\mathbf{R}_i^T\right] - \mathbf{x}_i\mathbf{x}_i^T \\
&= \frac{-(p-m)}{m(p-1)}\mathbf{x}_i\mathbf{x}_i^T + \frac{p(p-m)}{m(p-1)}\text{diag}\left(\mathbf{x}_i\mathbf{x}_i^T\right).
\end{aligned}$$

Thus, $\mathbb{E}[z_i^2] = (1/n^2)(\frac{p}{m}-1)(\mathbf{e}_j^T\mathbf{x}_i)^2$, and $\sigma^2 = \sum_{i=1}^n \mathbb{E}[z_i^2] \le (1/n^2)(\frac{p}{m}-1)\|\mathbf{X}\|_{\text{max-row}}^2$. Using the Bernstein inequality:

$$\mathbb{P}\{|u_j| \ge t\} \le 2\exp\left(\frac{-nt^2/2}{(\frac{p}{m}-1)\|\mathbf{X}\|_{\text{max-row}}^2/n + \tau(m,p)\|\mathbf{X}\|_{\text{max}}t/3}\right).$$

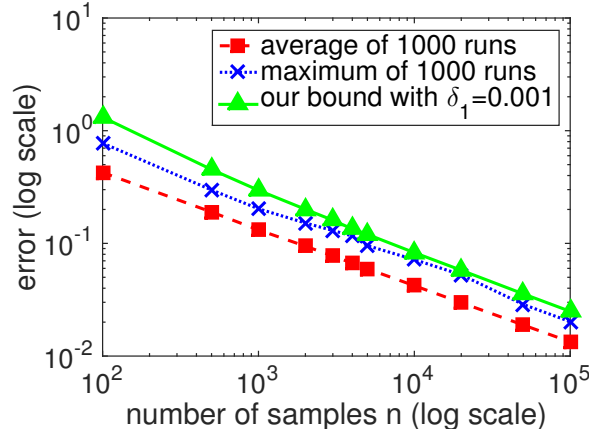Finally, we use the union bound over all p entries of $\mathbf{u}$. ∎

Figure 2. Verifying the sharpness of Thm. 4 on the synthetic data. For each n we report the average and maximum of the sample mean estimation error in 1000 runs compared with the theoretical error bound when $\delta_1 = 0.001$. The theoretical error bound is tight and decays exponentially as n increases.

Remark. *The result in Thm. 4 can be used to upper bound the error of our sample mean estimator $\widehat{\overline{\mathbf{x}}}_n$ in ℓ_2 norm. For all $t \geq 0$, with probability at least $1 - \delta_1$, where δ_1 defined in (10), we have:*

$$\frac{1}{\sqrt{p}} \|\widehat{\overline{\mathbf{x}}}_n - \overline{\mathbf{x}}_n\|_2 \leq t. \tag{15}$$

Remark. *Solving for t in (10) gives the following expression in terms of failure probability δ_1:*

$$t = \frac{1}{n}\left(\frac{\tau(m,p)}{3}\|\mathbf{X}\|_{max}\log\left(\frac{2p}{\delta_1}\right) + \sqrt{\left(\frac{\tau(m,p)}{3}\|\mathbf{X}\|_{max}\log\left(\frac{2p}{\delta_1}\right)\right)^2 + 2\left(\frac{p}{m}-1\right)\log\left(\frac{2p}{\delta_1}\right)\|\mathbf{X}\|_{max\text{-}row}^2}\right) \tag{16}$$

We consider a numerical experiment on synthetic data set to show the precision of Thm. 4. We set the parameters $p = 100$ and compression factor $\gamma = m/p = 0.3$ and consider 1000 runs for different values of n. Let $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denote a multivariate Gaussian distribution parameterized by its mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. In each run, we generate a set of n samples in $\mathbb{R}^p$ from the probabilistic generative model $\mathbf{x}_i = \overline{\mathbf{x}} + \epsilon_i$, where $\overline{\mathbf{x}}$ is a fixed vector drawn from the Gaussian distribution and the additive noise ϵ_i is drawn i.i.d. from $\mathcal{N}(\mathbf{0}, \mathbf{I}_p)$. We then keep $m = 30$ entries from each data sample uniformly at random without replacement to obtain a sparse matrix. Using Thm. 4, we find the estimates of the sample mean from the sparse matrix and compare with the actual ℓ_∞ error between the estimates and true sample means. Fig. 2 reports the average and maximum of 1000 runs for each value of n and compares that with the theoretical error bound t in (10) obtained by setting the failure probability $\delta_1 = 0.001$. The theoretical error bound is quite tight since it is close to the maximum of 1000 runs.

In Thm. 4, the failure probability δ_1 depends on the properties of the data set such as the maximum absolute value of $\mathbf{X}$. For a data matrix $\mathbf{X}$ with normalized columns, values of $\|\mathbf{X}\|_{\max}$ and $\|\mathbf{X}\|_{\max\text{-}row}^2$ can be relatively large due to the existence of large entries in $\mathbf{X}$ and, in the extreme case, we can get $\|\mathbf{X}\|_{\max} = 1$ and $\|\mathbf{X}\|_{\max\text{-}row}^2 = n$. Since both $\|\mathbf{X}\|_{\max}$ and $\|\mathbf{X}\|_{\max\text{-}row}^2$ are in the denominator, large values of these quantities work against the accuracy of the estimator and makes the failure probability δ_1 closer to 1. This fact motivates the use of preconditioning transformation discussed in Section III to smooth out large entries of $\mathbf{X}$ and reduce the values of $\|\mathbf{X}\|_{\max}$ and $\|\mathbf{X}\|_{\max\text{-}row}^2$.

Corollary 5. *In the setting of Thm. 4, assume $\mathbf{X}$ is preconditioned by the ROS (1) and $\gamma = m/p \leq 0.5$. By using the results of Corollary 2, we can find an upper bound for the failure probability δ_1 that holds with probability greater than 0.99:*

$$\delta_1 \leq 2p\exp\left(\frac{-mnt^2}{4/\eta\log(200np)(1+\sqrt{p}t/3)}\right), \tag{17}$$

and, thus, we can achieve high accuracy, e.g., $\delta_1 \leq 0.001$, for

$$m \geq \frac{1}{n} \cdot \frac{4}{\eta}\log(200np)\log(2000p)(t^{-2} + \frac{\sqrt{p}t^{-1}}{3}) \tag{18}$$

where $\eta = 1$ for $\mathbf{H}$ a Hadamard matrix and $\eta = 1/2$ for $\mathbf{H}$ a DCT matrix. Recall that t is the upper bound for estimation error in (11). To gain intuition and provide indicative values of m, we set $p = 512$, $\eta = 1$, and $t = 0.01$. Then, for example, the values of the lower bound in (18) are 137.2, 15.1, and 1.6 for $n = 10^5$, 10^6, and 10^7 respectively. Since m should be a natural number, we need to sample $m = 138$, $m = 16$, and $m = 2$ entries per data sample. This means that as the number of samples n increases, we can sample fewer entries per data sample, which makes our approach applicable to large-scale data sets.

To be formal, as n grows with p and t fixed, if the number of sub-sampled entries per data sample m is proportional to $\mathcal{O}(\log n / n)$, our sample mean estimator is accurate with high probability. Therefore, the amount of data we need to keep increases like $mn \propto \mathcal{O}(\log n)$.

V. THE COVARIANCE ESTIMATOR

In this section, we study the problem of covariance estimation for a set of data samples $\{\mathbf{x}_i\}_{i=1}^{n}$ from $\{\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\}_{i=1}^{n}$, where m out of p entries of each $\mathbf{x}_i$ are kept uniformly at random without replacement. We propose an unbiased estimator for the covariance matrix of the full data $\mathbf{C}_{\text{emp}} = 1/n \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T$ and study the closeness of our proposed covariance estimator to $\mathbf{C}_{\text{emp}}$ based on the spectral norm. Recall we do not impose structural assumptions on the covariance matrix such as $\mathbf{C}_{\text{emp}}$ being low-rank.

To begin, consider a rescaled version of the empirical covariance matrix of the sub-sampled data $\{\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\}_{i=1}^{n}$:

$$\widehat{\mathbf{C}}_{\text{emp}} := \frac{p(p-1)}{m(m-1)} \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{R}_i\mathbf{R}_i^T. \tag{19}$$

Based on Thm. B4, we can compute the expectation of $\widehat{\mathbf{C}}_{\text{emp}}$:

$$\mathbb{E}[\widehat{\mathbf{C}}_{\text{emp}}] = \mathbf{C}_{\text{emp}} + \frac{(p-m)}{(m-1)} \operatorname{diag}(\mathbf{C}_{\text{emp}}) \tag{20}$$

which consists of two terms, the covariance matrix of the full data $\mathbf{C}_{\text{emp}}$ (desired term) and an additional bias term that contains the elements on the main diagonal of $\mathbf{C}_{\text{emp}}$. However, as in [26], we can easily modify $\widehat{\mathbf{C}}_{\text{emp}}$ to obtain an unbiased estimator:

$$\widehat{\mathbf{C}}_n := \widehat{\mathbf{C}}_{\text{emp}} - \frac{(p-m)}{(p-1)} \operatorname{diag}(\widehat{\mathbf{C}}_{\text{emp}}) \tag{21}$$

where revisiting Thm. B4 shows that $\widehat{\mathbf{C}}_n$ is an unbiased estimator for $\mathbf{C}_{\text{emp}}$, i.e., $\mathbb{E}[\widehat{\mathbf{C}}_n] = \mathbf{C}_{\text{emp}}$. Next, we present a theorem to show the closeness of our proposed estimator $\widehat{\mathbf{C}}_n$ to the covariance matrix $\mathbf{C}_{\text{emp}}$.

Before stating the result, let us define $\mathbf{w}_i := \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i$, $i = 1, \ldots, n$, which is an m-sparse vector containing m entries of $\mathbf{x}_i$. We introduce a parameter $\rho > 0$ such that for all $i = 1, \ldots, n$ we have $\|\mathbf{w}_i\|_2^2 \leq \rho \|\mathbf{x}_i\|_2^2$. Obviously, $\|\mathbf{w}_i\|_2^2 \leq \|\mathbf{x}_i\|_2^2$ and for data sets with a few large entries, we can have $\|\mathbf{w}_i\|_2^2 = \|\mathbf{x}_i\|_2^2$ meaning that sub-sampling has not decreased the Euclidean norm. Therefore, $\rho \leq 1$ and we can always take $\rho = 1$. However, if we first apply the preconditioning transformation $\mathbf{HD}$ (1) to the data, we see that, with high probability, the sub-sampling operation decreases the Euclidean norm by a factor proportional to the compression factor $\gamma = m/p$. In fact, Corollary 3 with $\alpha = 1/100$ shows that $\rho = \frac{m}{p}\frac{2}{\eta}\log(200np)$ with probability greater than 0.99. As we will see, this is of great importance to decrease the variance of our covariance estimator and achieve high accuracy, which motivates using the preconditioning transformation before sub-sampling.

Theorem 6. *Let $\mathbf{C}_{emp}$ represent the covariance matrix of $\{\mathbf{x}_i\}_{i=1}^{n}$ and construct a rescaled version of the empirical covariance matrix from $\{\mathbf{w}_i = \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\}_{i=1}^{n}$, where each column of $\mathbf{R}_i \in \mathbb{R}^{p \times m}$ is chosen uniformly at random from the set of all canonical basis vectors without replacement:*

$$\widehat{\mathbf{C}}_{emp} = \frac{p(p-1)}{m(m-1)} \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{R}_i\mathbf{R}_i^T. \tag{22}$$

Let $\rho > 0$ be a bound such that for all $i = 1, \ldots, n$, we have $\|\mathbf{w}_i\|_2^2 \leq \rho \|\mathbf{x}_i\|_2^2$ (in particular, we can always take $\rho = 1$). Then,

$$\widehat{\mathbf{C}}_n = \widehat{\mathbf{C}}_{emp} - \frac{(p-m)}{(p-1)} \operatorname{diag}(\widehat{\mathbf{C}}_{emp}) \tag{23}$$

is an unbiased estimator for $\mathbf{C}_{emp}$, and for all $t \geq 0$,

$$\mathbb{P}\left\{ \|\widehat{\mathbf{C}}_n - \mathbf{C}_{emp}\|_2 \leq t \right\} \geq 1 - \delta_2, \quad \delta_2 = p \exp\left(\frac{-t^2/2}{\sigma^2 + Lt/3}\right) \tag{24}$$

where

$$L = \frac{1}{n}\left\{ \left(\frac{p(p-1)}{m(m-1)}\rho + 1 \right) \|\mathbf{X}\|_{max\text{-}col}^2 + \frac{p(p-m)}{m(m-1)} \|\mathbf{X}\|_{max}^2 \right\} \tag{25}$$
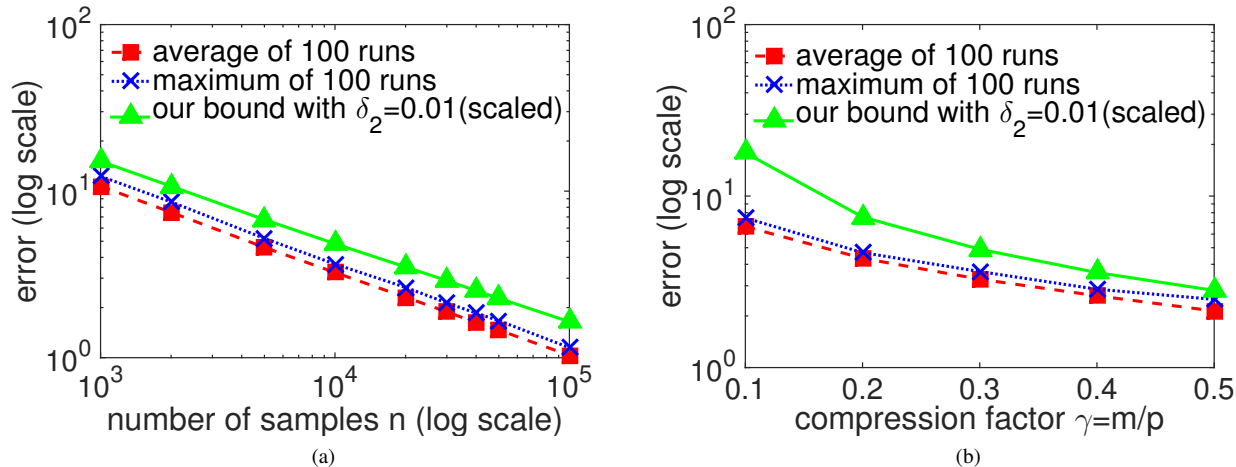
Figure 3. Verifying the accuracy of Thm. 6 on synthetic data. We set $p = 1000$ and plot the average and maximum of covariance estimation error in 100 runs for (a) varying n when $\gamma = 0.3$ fixed, and (b) varying γ when $n = 10p$ fixed. The empirical values are compared with the theoretical error bound for $\delta_2 = 0.01$ and scaled by a factor of 10. Our bounds are accurate to within an order of magnitude and they are representative of the empirical behavior of our covariance estimator in terms of n and γ.

and $\sigma^2 = \|\mathbb{E}[(\widehat{\mathbf{C}}_n - \mathbf{C}_{emp})^2]\|_2$ represents the variance:

$$\sigma^2 \leq \frac{1}{n} \left\{ \left(\frac{p(p-1)}{m(m-1)}\rho - 1 \right) \|\mathbf{X}\|^2_{max\text{-}col}\|\mathbf{C}_{emp}\|_2 \right.$$
$$+ \frac{p(p-1)(p-m)}{m(m-1)^2}\rho\|\mathbf{X}\|^2_{max\text{-}col}\| \operatorname{diag}(\mathbf{C}_{emp})\|_2$$
$$+ \frac{2p(p-1)(p-m)}{m(m-1)^2}\|\mathbf{X}\|^2_{max}\frac{\|\mathbf{X}\|^2_F}{n}$$
$$\left. + \frac{p(p-m)^2}{m(m-1)^2}\frac{\max_{j=1,\dots,p}\sum_{i=1}^n X^4_{j,i}}{n} \right\}. \tag{26}$$

Proof: The proof follows from the matrix Bernstein inequality (Thm. A3) and delayed to Appendix C. ∎

To gain some intuition and verify the accuracy of Thm. 6, we consider a numerical experiment. We set the parameter $p = 1000$ and show the accuracy of our proposed covariance estimator $\widehat{\mathbf{C}}_n$ for various numbers of samples n and compression factors γ. Fig. 3(a) shows the closeness of our covariance estimator $\widehat{\mathbf{C}}_n$ to the covariance matrix $\mathbf{C}_{emp}$, i.e., $\|\widehat{\mathbf{C}}_n - \mathbf{C}_{emp}\|_2$, over 100 runs for various values of n when $\gamma = m/p = 0.3$ is fixed. In each run, we generate a set of n data samples using the probabilistic generative model $\mathbf{x}_i = \sum_{j=1}^k \kappa_{ij}\lambda_j\mathbf{u}_j$, where we set $k = 5$ and $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_5]$ is a matrix of principal components with orthonormal columns obtained by performing QR decomposition on a $p \times k$ matrix with i.i.d. entries from $\mathcal{N}(0, 1)$. The coefficients κ_{ij} are drawn i.i.d. from $\mathcal{N}(0, 1)$ and the vector $\boldsymbol{\lambda}$ represents the energy of the data in each principal direction and we choose $\boldsymbol{\lambda} = (10, 8, 6, 4, 2)$. The empirical value of estimation error $\|\widehat{\mathbf{C}}_n - \mathbf{C}_{emp}\|_2$ is compared with the theoretical error bound t in (24) when the failure probability is chosen $\delta_2 = 0.01$, and the resulting error bound is scaled by a factor of 10.

Furthermore, we plot the empirical value of the estimation error vs. compression factor γ in Fig. 3(b) over 100 runs when a fixed number of samples $n = 10p$ are generated using the same generative model. As before, the empirical value is compared with the theoretical error bound when we choose $\delta_2 = 0.01$ and our bound is scaled by the same factor of 10. Our bounds are accurate to within an order of magnitude, and the theoretical result in Thm. 6 correctly captures the dependence of the estimation error $\|\widehat{\mathbf{C}}_n - \mathbf{C}_{emp}\|_2$ in terms of the parameters n and γ. For example, as n increases, the estimation error decreases exponentially for a fixed compression factor γ.

The other important consequence of Thm. 6 is revealing the connections between accuracy of our covariance estimator $\widehat{\mathbf{C}}_n$ and some properties of the data set $\mathbf{X}$ as well as the compression factor γ. Note that large values of parameters L and σ^2 work against the accuracy of $\widehat{\mathbf{C}}_n$ and make the failure probability δ_2 closer to 1, since they are in the denominator in (24). Let us assume that $\mathbf{X}$ has normalized columns so that $\|\mathbf{X}\|_{max\text{-}col} = 1$ and $\|\mathbf{X}\|^2_F = n$. With this normalization, $\|\mathbf{C}_{emp}\|_2 \leq 1$, and $\| \operatorname{diag}(\mathbf{C}_{emp})\|_2 \leq 1$ as well, which follows from exercise 27 in §3.3 [44] and $\mathbf{C}_{emp} \succeq 0$. In this case, both L and σ^2 scale as $\frac{1}{n}$ and the estimation error decreases exponentially as n increases for a fixed compression factor. Moreover, for a fixed n, $L \propto \mathcal{O}(\frac{p^2}{m^2})$ and $\sigma^2 \propto \mathcal{O}(\frac{p^3}{m^3})$. However, if we precondition the data $\mathbf{X}$ before sub-sampling as discussed in Section III, then $\rho \propto \mathcal{O}(\frac{m}{p})$ and the maximum absolute value of the entries $\|\mathbf{X}\|_{max}$ of the preconditioned data is proportional to $\frac{1}{\sqrt{p}}$ ignoring
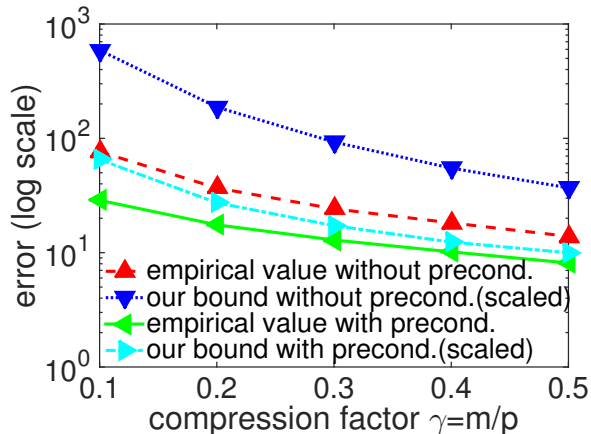
Figure 4. Effectiveness of preconditioning on the synthetic data set with few large entries. We plot the average of covariance estimation error over 100 runs for varying γ when $p = 512$ and $n = 1024$ in two cases of sub-sampling $\mathbf{X}$ (without preconditioning) and $\mathbf{Y} = \mathbf{HDX}$ (with preconditioning). We compare the empirical values with the theoretical error bound for $\delta_2 = 0.01$ and scaled by a factor 10 in these two cases. The preconditioning transformation $\mathbf{HD}$ leads to a noticeable reduction of estimation error in both empirical and theoretical results.

logarithmic factors. Thus, the leading term in L scales as $\mathcal{O}(\frac{p}{m})$ and the leading term in σ^2 scales as $\mathcal{O}(\frac{p^2}{m^2})$ which means that they are both reduced by a factor of $\frac{m}{p}$ under the preconditioning transformation. Specifically, simplifying (26) by dropping lower-order terms, assuming $p \gg m \gg 1$, using the normalization of $\mathbf{X}$ discussed above, as well as assuming preconditioning so $\rho = m/p$ and $\|\mathbf{X}\|_{\max}^2 \approx 1/p$, gives

$$\sigma^2 \lesssim \frac{1}{n} \left\{ \frac{p}{m} \|\mathbf{C}_{\text{emp}}\|_2 + \frac{p^2}{m^2} \| \operatorname{diag}(\mathbf{C}_{\text{emp}})\|_2 + \frac{2p^2}{m^3} + \frac{p}{m^3} \right\}. \tag{27}$$

Using just $\|\mathbf{C}_{\text{emp}}\|$, $\| \operatorname{diag}(\mathbf{C}_{\text{emp}})\| \leq 1$ then gives the bound $\sigma^2 \lesssim \mathcal{O}\left(\frac{1}{n} \frac{p^2}{m^2}\right)$, but this can be improved in special cases. Based on (27), we now consider tighter bounds for the few special cases, still assuming the data have been preconditioned so that $\mathbf{C}_{\text{emp}} = \frac{1}{n} \mathbf{HDXX}^T \mathbf{D}^T \mathbf{H}^T$:

1) If each $\mathbf{x}_i$ is a canonical basis vector chosen uniformly-at-random, then $\mathbf{C}_{\text{emp}} = \operatorname{diag}(\mathbf{C}_{\text{emp}}) = p^{-1}\mathbf{I}_p$ in the limit as $n \to \infty$. The third term in (27) dominates, and the bound is $1/n \cdot p^2/m^3$ which is quite strong.
2) If each entry of $\mathbf{X}$ is chosen i.i.d. $\mathcal{N}(0, 1/p)$, then the columns have unit norm in expectation and again $\mathbf{C}_{\text{emp}} \to p^{-1}\mathbf{I}_p$ as $n \to \infty$, so the bound is the same as the previous case (and preconditioning neither helps nor hurts).
3) If $\mathbf{x}_i = \mathbf{x}$ for all columns $i = 1, \ldots, n$ and for some fixed (and normalized) column $\mathbf{x}$, then $\mathbf{C}_{\text{emp}} = \mathbf{HDxx}^T \mathbf{D}^T \mathbf{H}^T$, so $\|\mathbf{C}_{\text{emp}}\|_2 = 1$. For example, if $\mathbf{x}$ is a canonical basis vector, then $\mathbf{C}_{\text{emp}}$ is the outer-product of a column of the Hadamard matrix, so $\operatorname{diag}(\mathbf{C}_{\text{emp}}) = p^{-1}\mathbf{I}_p$ and thus $\sigma^2 \lesssim 1/n \cdot p/m$, similar to [26]. As well as improving ρ and $\|\mathbf{X}\|_{\max}$, in this case preconditioning has the effect of spreading out the energy in $\mathbf{C}_{\text{emp}}$ away from the diagonal. Without preconditioning, then $\mathbf{C}_{\text{emp}} = \mathbf{xx}^T$, so if $\mathbf{x}$ is a standard basis vector, $\mathbf{C}_{\text{emp}} = \operatorname{diag}(\mathbf{C}_{\text{emp}})$. Intuitively, this is a bad case for non-preconditioned sampling, since there is a slim chance of sampling the only non-zero coordinate. Note that this perfectly correlated case is difficult for approaches based on $\mathbf{\Omega X}$ since each column is the same so the measurements are redundant, whereas our approach uses a unique sampling matrix for each column i and thus gives more information.

In order to clarify the discussion, we provide a simple numerical experiment on a synthetic data set with a few large entries. We then show the effectiveness of the preconditioning transformation $\mathbf{HD}$ on the accuracy of our covariance estimator $\widehat{\mathbf{C}}_n$ and, consequently, accuracy of the estimated principal components on this data set.

In our experiment, we set $p = 512$, and generate $n = 1024$ data samples from the same probabilistic generative model with $k = 10$ and $\mathbf{U} \in \mathbb{R}^{p \times k}$ containing 10 principal components chosen from the set of all canonical basis vectors, and $\boldsymbol{\lambda} = (10, 9, \ldots, 1)$. In Fig. 4, we plot the average of estimation error $\|\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}}\|_2$ over 100 runs for various values of the compression factor $\gamma = m/p$. As described in Section III, we can use a simple unitary transformation $\mathbf{HD}$ to precondition the data and smooth out large entries. Thus, we consider the case where the data is first preconditioned, i.e., $\mathbf{Y} = \mathbf{HDX}$. In this case the estimation error is $\|\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}}\|_2$, where $\mathbf{C}_{\text{emp}} = \frac{1}{n}\mathbf{YY}^T$, and is also plotted in Fig. 4. Moreover, we report the theoretical error bounds when we choose $\delta_2 = 0.01$ and scale our bounds by the same factor of 10. The preconditioning decreases the error by almost a factor of 2, both in experiment and via the theoretical bounds.

To show the importance of this error reduction in covariance estimation, we find the number of recovered principal components obtained from the eigendecomposition of $\widehat{\mathbf{C}}_n$ for both cases. After finding the first $k = 10$ eigenvectors of $\widehat{\mathbf{C}}_n$, we compute the inner product magnitude between the recovered and true principal components and we declare that a principal component is "recovered" if the corresponding inner product magnitude is greater than 0.95. The mean and standard deviation of the number of recovered principal components for varying compression factors $\gamma = m/p$ are reported in Table I. As we see, the

Table I

NUMBER OF RECOVERED PRINCIPAL COMPONENTS FOR VARIOUS VALUES OF COMPRESSION FACTOR $\gamma = m/p$ OVER 100 RUNS

γ	(without preconditioning)		(with preconditioning)	
	mean	standard deviation	mean	standard deviation
0.1	0.98	0.99	5.12	0.40
0.2	3.53	1.76	7.01	0.10
0.3	6.85	1.67	8.00	0
0.4	8.18	1.58	8.42	0.49
0.5	9.31	1.03	9.00	0

preconditioning transformation leads to a significant gain in accuracy of the estimated principal components, especially for small values of the compression factor, which are of great importance for the big data and distributed data settings. Additionally, the variance in the estimate is much reduced across the whole range of γ.

VI. SPARSIFIED K-MEANS CLUSTERING

Clustering is a commonly used unsupervised learning task which refers to identifying clusters of similar data samples in a data set. The K-means algorithm [34] is one of the most popular hard clustering algorithms that has been used in many fields such as data mining and machine learning.

Despite its simplicity, running K-means on large-scale data sets presents new challenges and considerable efforts have been made to introduce memory/computation efficient clustering algorithms. In this paper, we present a variant of the K-means algorithm which allows us to find a set of cluster centers as well as assignment of the data. The idea is to precondition and sample the data in one pass over the data to achieve a sparse matrix, therefore reducing processing time and saving memory, and applicable to streaming and distributed data.

First, we review the standard K-means algorithm. Given a data set $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$, the goal of K-means is to partition the data into a known number of K clusters such that $\boldsymbol{\mu}_k \in \mathbb{R}^p$ is the prototype associated with the k-th cluster for $k = 1, \ldots, K$. We also introduce a set of binary indicator variables $c_{ik} \in \{0, 1\}$ to represent the assignments, where $\mathbf{c}_i = [c_{i1}, \ldots, c_{iK}]^T$ is the k-th canonical basis vector in $\mathbb{R}^K$ if and only if $\mathbf{x}_i$ belongs to the k-th cluster.

Let us define cluster centers $\boldsymbol{\mu} = \{\boldsymbol{\mu}_k\}_{k=1}^K$ and the assignments of the data samples $\mathbf{c} = \{\mathbf{c}_i\}_{i=1}^n$. The K-means algorithm attempts to minimize the sum of the squared Euclidean distances of each data point to its assigned cluster:

$$J(\mathbf{c}, \boldsymbol{\mu}) = \sum_{i=1}^n \sum_{k=1}^K c_{ik} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2. \tag{28}$$

The objective $J(\mathbf{c}, \boldsymbol{\mu})$ is minimized by an iterative algorithm that (step one) updates assignments $\mathbf{c}$ and (step two) updates $\boldsymbol{\mu}$ as follows:

Step 1: Minimize $J(\mathbf{c}, \boldsymbol{\mu})$ over c, keeping $\boldsymbol{\mu}$ fixed:

$$\forall \ i = 1, \ldots, n : \ c_{ik} = \begin{cases} 1, & k = \arg\min_j \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2^2 \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

Step 2: Minimize $J(\mathbf{c}, \boldsymbol{\mu})$ over c, keeping $\boldsymbol{\mu}$ fixed:

$$\forall \ k = 1, \ldots K : \ \boldsymbol{\mu}_k = \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} \mathbf{x}_i \tag{30}$$

where $\mathcal{I}_k$ denotes the set of samples assigned to the k-th cluster in Step 1 and $n_k = |\mathcal{I}_k|$. Therefore, the update formula for cluster center $\boldsymbol{\mu}_k$ is the sample mean of the data samples in $\mathcal{I}_k$. To initialize K-means, a set of cluster centers can be chosen uniformly at random from the data set $\mathbf{X}$. However, we use the recent K-means++ algorithm [45] to choose the initial cluster centers since it improves the performance of K-means over the worst-case random initializations.

Next, we consider a probabilistic mixture model to find an optimal objective function for our sparsified K-means clustering algorithm using Maximum-Likelihood (ML) estimation.

A. Optimal Objective Function via ML Estimation

One appealing aspect of the K-means algorithm is that the objective function (28) coincides with the log-likelihood function of a mixture of K Gaussian components (clusters), with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma} = \lambda \mathbf{I}_p$, where $\lambda > 0$ is fixed or "known" (its actual value is unimportant), and treating $\mathbf{c}$ and $\boldsymbol{\mu}$ as unknown parameters. We show that under the same assumptions, K-means clustering on sampled preconditioned data enjoys the same ML interpretation.

Let $p(\mathbf{x}_i|\boldsymbol{\mu}, \mathbf{c}_i)$ denote the conditional probability distribution of sample $\mathbf{x}_i$ given that a set of centers $\boldsymbol{\mu}$ and a particular value of $\mathbf{c}_i$ are known. Under this setup, the conditional distribution of $\mathbf{x}_i$ is Gaussian and it can be written as follows:

$$p\left(\mathbf{x}_i|\boldsymbol{\mu}, \mathbf{c}_i\right) = \prod_{k=1}^{K} p\left(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}\right)^{c_{ik}} \tag{31}$$

where

$$p\left(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}\right) = \frac{1}{(2\pi\lambda)^{\frac{p}{2}}} \exp\left(-\frac{1}{2\lambda}\left\|\mathbf{x}_i - \boldsymbol{\mu}_k\right\|_2^2\right).$$

Given that $\mathbf{x}_i$ belongs to the k-th cluster with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma} = \lambda\mathbf{I}_p$, then the preconditioned data $\mathbf{y}_i = \mathbf{HD}\mathbf{x}_i$ also has a Gaussian distribution with mean $\boldsymbol{\mu}'_k := \mathbb{E}[\mathbf{y}_i] = \mathbf{HD}\boldsymbol{\mu}_k$ and the same covariance $\boldsymbol{\Sigma} = \lambda\mathbf{I}_p$ because $\mathbf{H}$ and $\mathbf{D}$ are orthonormal matrices, i.e., $(\mathbf{HD})(\mathbf{HD})^T = \mathbf{I}_p$. Note that we can also find $\boldsymbol{\mu}_k$ from $\boldsymbol{\mu}'_k$ using the equation:

$$\boldsymbol{\mu}_k = (\mathbf{HD})^T \boldsymbol{\mu}'_k. \tag{32}$$

We now take n independent *realizations* of the sampling matrix $\mathbf{R}$, denoted $\mathbf{R}_1, \ldots, \mathbf{R}_n$, each consisting of m canonical basis vectors. Then, given that $\mathbf{y}_i$ belongs to the k-th cluster with mean $\boldsymbol{\mu}'_k$ and covariance matrix $\boldsymbol{\Sigma} = \lambda\mathbf{I}_p$, the sub-sampled data $\mathbf{z}_i = \mathbf{R}_i^T\mathbf{y}_i$ also has a Gaussian distribution with mean $\mathbb{E}[\mathbf{z}_i] = \mathbf{R}_i^T\boldsymbol{\mu}'_k$ and covariance $\lambda\mathbf{I}_m$, since $\mathbf{R}_i^T\mathbf{R}_i = \mathbf{I}_m$ based on Thm. B4. Hence

$$p\left(\mathbf{z}_i|\boldsymbol{\mu}'_k, \boldsymbol{\Sigma}\right) = \frac{1}{(2\pi\lambda)^{\frac{m}{2}}} \exp\left(-\frac{1}{2\lambda}\left\|\mathbf{z}_i - \mathbf{R}_i^T\boldsymbol{\mu}'_k\right\|_2^2\right)$$

and, thus, we have the following expression for the conditional distribution of $\mathbf{z}_i$:

$$p\left(\mathbf{z}_i|\boldsymbol{\mu}', \mathbf{c}_i\right) = \prod_{k=1}^{K} p\left(\mathbf{z}_i|\boldsymbol{\mu}'_k, \boldsymbol{\Sigma}\right)^{c_{ik}}. \tag{33}$$

Next, we consider ML estimation when we have access to the sampled preconditioned data $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n]$:

$$p\left(\mathbf{Z}|\boldsymbol{\mu}', \mathbf{c}\right) = \prod_{i=1}^{n}\prod_{k=1}^{K} p\left(\mathbf{z}_i|\boldsymbol{\mu}'_k, \boldsymbol{\Sigma}\right)^{c_{ik}}$$

and taking the logarithm of the likelihood function:

$$\log p(\mathbf{Z}|\boldsymbol{\mu}', \mathbf{c}) = -\frac{mn}{2}\log(2\pi\lambda) - \frac{1}{2\lambda}\sum_{i=1}^{n}\sum_{k=1}^{K} c_{ik}\left\|\mathbf{z}_i - \mathbf{R}_i^T\boldsymbol{\mu}'_k\right\|_2^2.$$

Hence, the ML estimate of the unknown parameters $\mathbf{c}$ and $\boldsymbol{\mu}'$ (or equivalently $\boldsymbol{\mu}$) is obtained by minimizing:

$$J'(\mathbf{c}, \boldsymbol{\mu}') = \sum_{i=1}^{n}\sum_{k=1}^{K} c_{ik}\left\|\mathbf{z}_i - \mathbf{R}_i^T\boldsymbol{\mu}'_k\right\|_2^2. \tag{34}$$

Note that $J'(\mathbf{c}, \boldsymbol{\mu}')$ can be written as:

$$J'(\mathbf{c}, \boldsymbol{\mu}') = \sum_{i=1}^{n}\sum_{k=1}^{K} c_{ik}\left\|\mathbf{R}_i^T\left(\mathbf{y}_i - \boldsymbol{\mu}'_k\right)\right\|_2^2 \tag{35}$$

and for $\mathbf{R}_i = \mathbf{I}_p$, $i = 1, \ldots, n$, it reduces to the objective function of the standard K-means (28) because the preconditioning transformation $\mathbf{HD}$ is an orthonormal matrix.

B. The Sparsified K-means Algorithm

Similar to the K-means algorithm, we minimize the objective function $J'(\mathbf{c}, \boldsymbol{\mu}')$ in an iterative procedure that (step one) updates assignments $\mathbf{c}$ and (step two) updates $\boldsymbol{\mu}'$:

Step 1: Minimize $J'(\mathbf{c}, \boldsymbol{\mu}')$ holding $\boldsymbol{\mu}'$ fixed:

In (34), the terms involving different n are independent and we assign each sampled preconditioned data $\mathbf{z}_i = \mathbf{R}_i^T\mathbf{y}_i \in \mathbb{R}^m$ to the closest cluster:

$$\forall\ i = 1, \ldots, n: \ c_{ik} = \begin{cases} 1, & k = \arg\min_j \left\|\mathbf{z}_i - \mathbf{R}_i^T\boldsymbol{\mu}'_j\right\|_2^2 \\ 0, & \text{otherwise} \end{cases} \tag{36}$$

The connection between this step and the first step of K-means is immediate mainly due to the Johnson-Lindenstrauss (JL) lemma. In fact, the accuracy of this step depends on the preservation of the Euclidean norm under selecting m entries of a p-dimensional vector. Based on the well-known fast JL transform [4], one needs to first smooth out data samples with a few

large entries to ensure the preservation of the Euclidean norm with high probability. In particular, the author in [43] showed that selecting m entries of the preconditioned data uniformly at random without replacement preserves the geometry of the data as well as the Euclidean norm. A direct consequence of this result stated in Thm. D6 shows that pairwise distances between each point and cluster centers are preserved.

Step 2: Minimize $J'(\mathbf{c}, \boldsymbol{\mu}')$ holding $\mathbf{c}$ fixed:

Given the assignments $\mathbf{c}$ from Step 1, we can write (34) as:

$$J'(\boldsymbol{\mu}') = \sum_{i \in \mathcal{I}_k} \left\| \mathbf{R}_i^T \left(\mathbf{y}_i - \boldsymbol{\mu}_k' \right) \right\|_2^2 \tag{37}$$

where $\mathcal{I}_k$ represents the set of samples assigned to the k-th cluster and recall that $n_k = |\mathcal{I}_k|$. The terms involving different k are independent and each term is a quadratic function of $\boldsymbol{\mu}_k'$. Thus, each term can be minimized individually by setting its derivative with respect to $\boldsymbol{\mu}_k'$ to zero giving:

$$\left(\sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T \right) \boldsymbol{\mu}_k' = \sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T \mathbf{y}_i. \tag{38}$$

Note that $\sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T \in \mathbb{R}^{p \times p}$ is a diagonal matrix, where its j-th diagonal element counts the number of cases the j-th canonical basis vector is chosen in the sampling matrices $\mathbf{R}_i$ for all $i \in \mathcal{I}_k$, denoted by $n_k^{(j)}$. Therefore, for any j with $n_k^{(j)} = 0$, we cannot estimate the j-th entry of $\boldsymbol{\mu}_k'$ and the corresponding entry should be removed from (38). Given that $n_k^{(j)} > 0$ for all j, $\boldsymbol{\mu}_k'$ is updated as follows:

$$\boldsymbol{\mu}_k' = \mathrm{diag}\left(\left[\frac{1}{n_k^{(1)}}, \ldots, \frac{1}{n_k^{(p)}} \right] \right) \left(\sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T \mathbf{y}_i \right). \tag{39}$$

Recall that each $\mathbf{R}_i \mathbf{R}_i^T \mathbf{y}_i$ is the sampled preconditioned data such that m out p entries are kept uniformly at random. Hence, the update formula for $\boldsymbol{\mu}_k'$ is the *entry-wise sample mean* of the sparse data samples in the k-th cluster. The sparsified K-means algorithm is summarized in Algorithm 1.

Algorithm 1 Sparsified K-means

Input: Dataset $\mathbf{X} \in \mathbb{R}^{p \times n}$, number of clusters K, compression factor $\gamma = \frac{m}{p} < 1$.

Output: Assignments $\mathbf{c} = \{\mathbf{c}_i\}_{i=1}^n \in \mathbb{R}^K$, cluster centers $\boldsymbol{\mu} = \{\boldsymbol{\mu}_k\}_{k=1}^K \in \mathbb{R}^p$.

  1: **function** SPARSIFIED K-MEANS($\mathbf{X}, K, \gamma$)
  2:     $\mathbf{Y} \leftarrow \mathbf{HDX}$ ▷ See Eq. (1)
  3:     **for** $i = 1, \ldots, n$ **do**
  4:         $\mathbf{w}_i = \mathbf{R}_i \mathbf{R}_i^T \mathbf{y}_i$ ▷ $\mathbf{R}_i \in \mathbb{R}^{p \times m}$:sampling matrix
  5:     Find initial cluster centers via K-means++ [45]
  6:     **for** each iteration **do**
  7:         update assignments $\mathbf{c}$ ▷ See Eq. (36)
  8:         update cluster centers $\boldsymbol{\mu}'$ ▷ See Eq. (39)
  9:     $\boldsymbol{\mu} = (\mathbf{HD})^T \boldsymbol{\mu}'$ ▷ See Eq. (32)
10:     Return $\mathbf{c}$ and $\boldsymbol{\mu}$.

Now, we return to equation (38) and study the accuracy of the estimated solution $\boldsymbol{\mu}_k'$. To do this, we re-write (38) as:

$$\mathbf{H}_k \boldsymbol{\mu}_k' = \mathbf{m}_k \tag{40}$$

where

$$\mathbf{H}_k = \frac{p}{m} \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T, \ \mathbf{m}_k = \frac{p}{m} \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T \mathbf{y}_i. \tag{41}$$

Next, we show that $\mathbf{H}_k$ converges to the identity matrix $\mathbf{I}_p$ as the number of samples in each cluster n_k increases.

Theorem 7. *Consider $\mathbf{H}_k$ defined in (41). Then, for all $t \geq 0$:*

$$\mathbb{P}\left\{ \|\mathbf{H}_k - \mathbf{I}_p\|_2 \leq t \right\} \geq 1 - \delta_3 \tag{42}$$

where the failure probability,

$$\delta_3 = p \exp\left(\frac{-n_k t^2 / 2}{\left(\frac{p}{m} - 1\right) + \left(\frac{p}{m} + 1\right) t / 3} \right). \tag{43}$$
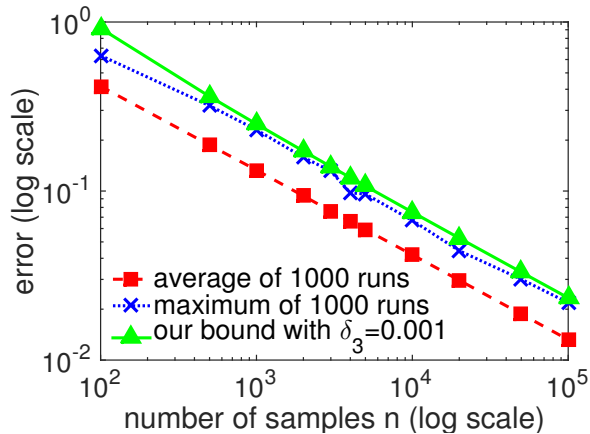
Figure 5. Verifying the accuracy of Thm. 7. We set parameters $p = 100$ and $\gamma = m/p = 0.3$ and plot the average and maximum of $\|\mathbf{H}_k - \mathbf{I}_p\|_2$ over 1000 runs for varying n. We compare the empirical values with our theoretical bound when $\delta_3 = 0.001$. We see that our bound is tight.

Proof: We can write $\mathbf{S} = \mathbf{H}_k - \mathbf{I}_p = \sum_{i=1}^{n_k} \mathbf{Z}_i$, where

$$\mathbf{Z}_i = \frac{1}{n_k} \left(\frac{p}{m} \mathbf{R}_i \mathbf{R}_i^T - \mathbf{I}_p \right), \ i = 1, \ldots, n_k$$

are independent and symmetric random matrices. Moreover, we have $\mathbb{E}[\mathbf{Z}_i] = \mathbf{0}$ using Thm. B4. To apply the matrix Bernstein inequality given in Appendix A, we should find a uniform bound on the spectral norm of each summand $\|\mathbf{Z}_i\|_2$:

$$\|\mathbf{Z}_i\|_2 \leq \frac{1}{n_k} \left(\left\| \frac{p}{m} \mathbf{R}_i \mathbf{R}_i^T \right\|_2 + \|\mathbf{I}_p\|_2 \right) = \frac{1}{n_k} \left(\frac{p}{m} + 1 \right) \tag{44}$$

where it follows from the triangle inequality for the spectral norm and the fact that $\mathbf{R}_i \mathbf{R}_i^T \in \mathbb{R}^{p \times p}$ is a diagonal matrix with only m ones on the diagonal and the rest equal to zero.

Next, we find $\mathbb{E}[\mathbf{Z}_i^2]$ using the results of Thm. B4:

$$\mathbb{E}\left[\mathbf{Z}_i^2\right] = \frac{1}{n_k^2} \mathbb{E}\left[\left(\frac{p^2}{m^2} - 2\frac{p}{m} \right) \mathbf{R}_i \mathbf{R}_i^T + \mathbf{I}_p \right] = \frac{1}{n_k^2} \left(\frac{p}{m} - 1 \right) \mathbf{I}_p$$

and thus

$$\sigma^2 = \left\| \sum_{i=1}^{n_k} \mathbb{E}\left[\mathbf{Z}_i^2\right] \right\|_2 = \frac{1}{n_k} \left(\frac{p}{m} - 1 \right). \tag{45}$$

We now use Theorem A3 and this completes the proof. ■

To verify the accuracy of Thm. 7, we consider a numerical experiment. We set the parameters $p = 100$ and compression factor $\gamma = m/p = 0.3$ and show the closeness of $\mathbf{H}_k$ to $\mathbf{I}_p$ for various values of n over 1000 runs. For each value of n, we generate n sampling matrices $\mathbf{R}_i$ consisting of m distinct canonical basis vectors uniformly at random. We report the average and maximum of empirical values $\|\mathbf{H}_k - \mathbf{I}_p\|_2$ over 1000 runs in Fig. 5. We also compare the empirical values with our theoretical error bound t in (43), when the failure probability $\delta_3 = 0.001$. We see that our error bound is tight and very close to the maximum of 1000 since $\delta_3 = 0.001$.

Now, we present a theorem to show the connection between the updated cluster center in our sparsified K-means in (40) and the update formula for the standard K-means algorithm.

Theorem 8. *Consider the update formula for the k-th cluster center in our sparsified K-means algorithm $\boldsymbol{\mu}_k = (\mathbf{HD})^T \boldsymbol{\mu}_k'$, where $\boldsymbol{\mu}_k'$ is given by the equation $\mathbf{H}_k \boldsymbol{\mu}_k' = \mathbf{m}_k$ in (40). Let $\overline{\mathbf{x}}_k$ denote the sample mean of the data samples in the k-th cluster, i.e., $\overline{\mathbf{x}}_k = \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} \mathbf{x}_i$, which is the standard update formula for K-means on $\mathbf{X}$. Then, for all $t \geq 0$,*

$$\frac{1}{\sqrt{p}} \|\boldsymbol{\mu}_k - \overline{\mathbf{x}}_k\|_2 \leq t \left(1 + \frac{1}{\sqrt{p}} \|\boldsymbol{\mu}_k\|_2 \right) \tag{46}$$

with probability greater than $1 - \max\{\delta_1, \delta_3\}$, where

$$\delta_1 = 2p \exp\left(\frac{-n_k t^2/2}{(\frac{p}{m} - 1)\|\mathbf{Y}\|_{max\text{-}row}^2/n + \tau(m,p)\|\mathbf{Y}\|_{max} t/3} \right), \tag{47}$$

$\tau(m,p)$ is defined in (9) and δ_3 is given in (43). Recall that $\mathbf{Y} = \mathbf{HDX}$ is the preconditioned data in our sparsified K-means algorithm.

Table II
LOW-PASS ALGORITHMS FOR K-MEANS CLUSTERING

| Algorithm | Passes through data... | |
	...to find $\boldsymbol{\mu}$	...to find $\mathbf{c}$
Sparsified K-means (1-pass)	1	1
Sparsified K-means (2-pass)	2	2
Feature extraction	2	1
Feature selection	4	3

Proof: Based on Thm. 4 and Thm. 7, we re-write equation (40) as:

$$\left(\mathbf{I}_p + \mathbf{E}\right) \boldsymbol{\mu}'_k = \mathbf{HD}\overline{\mathbf{x}}_k + \mathbf{e}$$

where $\|\mathbf{E}\|_2 \leq t$ and $\|\mathbf{e}\|_2 \leq \sqrt{p}t$ with probabilities greater than $1 - \delta_3$ and $1 - \delta_1$ respectively. Thus, with probability greater than $1 - \max\{\delta_1, \delta_3\}$, we have:

$$
\begin{aligned}
\|\boldsymbol{\mu}'_k - \mathbf{HD}\overline{\mathbf{x}}_k\|_2 &= \|\mathbf{e} - \mathbf{E}\boldsymbol{\mu}'_k\|_2 \\
&\leq \|\mathbf{e}\|_2 + \|\mathbf{E}\boldsymbol{\mu}'_k\|_2 \leq \|\mathbf{e}\|_2 + \|\mathbf{E}\|_2\|\boldsymbol{\mu}'_k\|_2 \\
&\leq t\sqrt{p}\left(1 + \frac{\|\boldsymbol{\mu}'_k\|_2}{\sqrt{p}}\right)
\end{aligned}
\tag{48}
$$

where we used the triangle inequality for the spectral norm. Recall that $\mathbf{HD}$ is an orthonormal matrix and $\boldsymbol{\mu}'_k = \mathbf{HD}\boldsymbol{\mu}_k$. Thus, $\|\boldsymbol{\mu}'_k - \mathbf{HD}\overline{\mathbf{x}}_k\|_2 = \|\boldsymbol{\mu}_k - \overline{\mathbf{x}}_k\|_2$ and $\|\boldsymbol{\mu}'_k\|_2 = \|\boldsymbol{\mu}_k\|_2$ and this completes the proof. ∎

VII. NUMERICAL EXPERIMENTS

We implement the sparsified K-means algorithm in a mixture of Matlab and C, available online[3]. Since K-means attempts to minimize a non-convex objective, the starting points have a large effect. We use the recent K-means++ algorithm [45] for choosing starting points, and re-run the algorithm for different sets of starting points and then choose the results with the smallest objective value. All results except the big-data tests use 20 different starting trials.

Timing results are from running the algorithm on a desktop computer with two Intel Xeon EF-2650 v3 CPUs at 2.4–3.2 GHz and 8 cores and 20 MB cache each, and should be interpreted carefully. First, we note that K-means is iterative and so the number of iterations may change slightly depending on the variant. Furthermore, none of the code was optimized for small problems, so timing results under about 10 seconds do not scale with n and p as they do at large scale. At the other extreme, our first series of tests are not on out-of-core data, so the benefits of a single-pass algorithm are not apparent. Our subsequent tests are out-of-core implementations, meaning that they explicitly load data from the hard drive to RAM as few times as possible, and so the number of passes through the data becomes relevant, cf. Table II.

We also caution about interpreting the accuracy results for correct identification of clusters. In our experience, if the accuracy is greater than about 75%, then using the result as the initial value for a single-step of standard K-means, thus increasing the number of passes through the data by one, is sufficient to match the accuracy of the standard K-means algorithm.

A two-pass sparsified K-means algorithm can be constructed by running the one-pass sparsified K-means in Algorithm 1 to compute the assignments as well as the cluster centers, then re-computing the cluster center estimates $\boldsymbol{\mu}$ as the average of their assigned data points in the original (non-sampled) domain. Meanwhile, we can re-assign the data samples to the cluster centers in the original domain based on the previous center estimates from one-pass sparsified K-means. The same extra-pass procedure *must* be applied to feature extraction and feature selection, since their default center estimates $\boldsymbol{\mu}$ are in a compressed domain.

Our tests compare with the feature selection and feature extraction algorithms of [36]. In *feature selection*, one first uses a fast approximate SVD algorithm, e.g., [7], [46], to compute an estimate of the left singular vectors of the data matrix $\mathbf{X}$. Then, the selection of m rows of $\mathbf{X}$ is done with a randomized sampling approach with probabilities that are computed from the estimated left singular vectors. This can be written as $\mathbf{\Omega}\mathbf{X}$, where rows of the sampling matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times p}$ are chosen from the set of canonical basis vectors in $\mathbb{R}^p$ based on the computed probabilities. In *feature extraction*, the samples are again $\mathbf{\Omega}\mathbf{X}$ but the sampling matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times p}$ is set to be a random sign matrix. Thus, the computational cost of feature extraction is dominated by the matrix-matrix multiplication $\mathbf{\Omega}\mathbf{X}$, whereas the dominant cost in feature selection is the approximate SVD.

A. Sketched K-means for faster computation

Sampling the data leads to both computational time and memory benefits, with computational time benefits becoming more significant for more complicated algorithms, such as the Expectation-Maximization algorithm in Gaussian mixture models that

---

[3]https://github.com/stephenbeckr/SparsifiedKMeans

Algorithm 2 Sparsified K-means, 2-pass
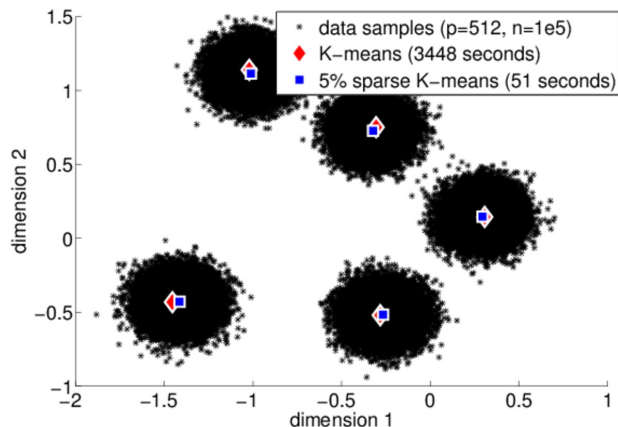

1: **function** SPARSIFIED K-MEANS 2-PASS($\mathbf{X}, K, \gamma$)
2:     $(\hat{\mathbf{c}}, \hat{\boldsymbol{\mu}})$ = SPARSIFIED K-MEANS($\mathbf{X}, K, \gamma$) $\triangleright$ Alg. 1
3:     **for** $k = 1, \ldots, K$ **do**
4:         $\boldsymbol{\mu}_k = \mathbf{0}$, $\mathcal{I}_k = \emptyset$
5:     **for** $i = 1, \ldots, n$ **do**
6:         Find cluster assignment, i.e., k s.t. $\hat{\mathbf{c}}_i = \mathbf{e}_k$
7:         $\boldsymbol{\mu}_k \mathrel{+}= \mathbf{x}_i$, $\mathcal{I}_k = \mathcal{I}_k \cup \{i\}$
8:         $\mathbf{c}_i = \arg\min_{k=1,\ldots,K} \|\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k\|_2^2$
9:     **for** $k = 1, \ldots, K$ **do**
10:         $\boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k / |\mathcal{I}_k|$
11:     Return $\mathbf{c}$ and $\boldsymbol{\mu}$.

Figure 6.   Standard K-means and our sparse version of K-means, on synthetic data, $n = 10^5$.

require eigenvalue decompositions. Even for the standard K-means algorithm, sub-sampling leads to faster computation. The most expensive computation in K-means is finding the nearest cluster center for each point, since a naive implementation of this costs $\mathcal{O}(pnK)$ flops per iteration[4]. By effectively reducing the dimension from p to m, the sparse version sees a speedup of roughly $\gamma^{-1} = p/m$.

Fig. 6 demonstrates this speedup experimentally on a toy problem. The data of size $p = 512$ and $n = 10^5$ are generated artificially so that each point belongs to one of $K = 5$ clusters and is perturbed by a small amount of Gaussian noise. An optimized variant of Matlab's kmeans algorithm takes 3448 seconds to run.

We compare this with random Hadamard mixing followed by 5% sub-sampling, which takes 51 seconds. The first two dimensions of the data are shown in Fig. 6 which makes it clear that there is no appreciable difference in quality, while our sparsified K-means algorithm is approximately 67 times faster.

B. Comparison with dimensionality-reducing approaches on real data

For a realistic clustering application, experiments are performed on the MNIST dataset[5] which consists of centered versions of hand-written digits, each digit stored as a 28×28 pixel image. For processing, the images are vectorized so $p = 28^2 = 784$. The dataset includes both testing and training sets, though for our purposes we combined the two and report in-sample error, since the effect of sampling and dimensionality reduction to out-of-sample error is beyond our scope.

For simplicity of interpreting results, we use data from the samples of three digits ("0", "3" and "9"), so $K = 3$ in the clustering algorithm. There were 6903, 7141 and 6958 examples of each class of image, respectively, so $n = 21002$. The original data provides a ground-truth label, against which we compute accuracy by computing the total number of correctly assigned images, normalized by the total number of images. All the algorithms, except standard K-means, are stochastic, so we re-run the clustering 50 times and record the mean and standard deviation of these 50 trials. Recall that within each run, we choose the best of 20 random starting points.

a) Timing and accuracy: Clustering accuracy as a function of γ is shown in Fig. 7, which suggests sparsified K-means is the most accurate of the efficient methods, and that accuracy is further improved in the two-pass version. Timing results are

---

[4] We do not consider the variants of K-means based on kd-trees since these have running time exponential in p and are suitable for $p \lesssim 20$ [47].
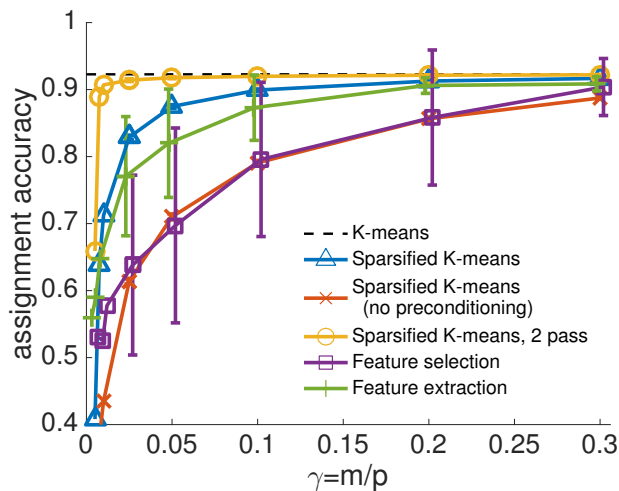[5] http://yann.lecun.com/exdb/mnist/

Figure 7. Accuracy of various K-means algorithms on the MNIST data, 50 trials. The plot shows the mean and the standard deviation error bars, empirically suggesting that feature-based algorithms show higher variance than the sampling-based algorithm. For example, at $\gamma = 0.1$, the standard deviation for sparsified K-means, sparsified K-means without preconditioning, 2-pass sparsified K-means, feature selection, and feature extraction, is 0.002, 0.004, 0.001, 0.1151 and 0.049, respectively. Moreover, the accuracy of 2-pass sparsified K-means reaches the accuracy of standard K-means even for small values of the compression factor γ. For visual clarity, we did not include the standard deviation error bars for $\gamma < 0.025$.
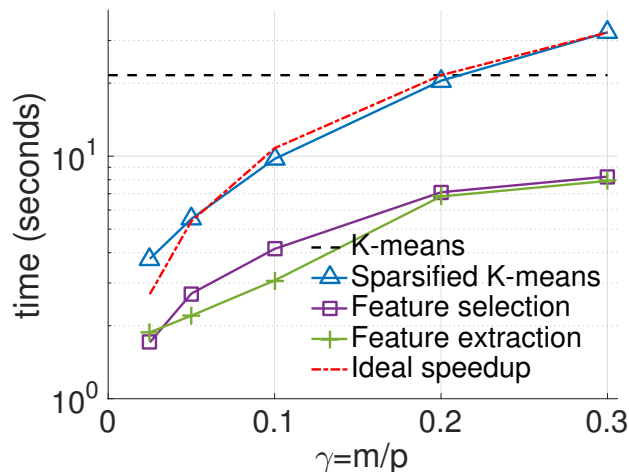

Figure 8. Timing of various K-means algorithms on the MNIST data. The three variants of sparsified K-means (with and without preconditioning, and 2-pass) all take approximately the same time on this dataset, so we only show the time for preconditioned sparsified K-means. The red dashed curve is proportional to γ, which is the ideal speedup ratio; the constant of proportionality is 5, chosen to make the curve line up with the sparsified K-means performance. Note that time is in log-scale; at $\gamma = 0.3$, sparsified K-means takes about 12 seconds while feature extraction takes 8 seconds.

shown in Fig. 8, which also shows the time for our optimized implementation of K-means on the full data. All the efficient algorithms show a speedup over K-means proportional to γ^{-1}, as expected, until the sparsity is near 5%, at which point various fixed costs in the computation start to dominate; one would expect ideal speedup to continue to lower γ if n were larger, cf. Table V. Sparsified K-means takes longer than standard K-means as $\gamma \to 1$ since it is inefficient to work with a sparse matrix format when the matrices are not actually sparse.

b) Center estimation: The estimated cluster centers $\boldsymbol{\mu}$ from several low-pass K-means algorithms are shown in Fig. 9, for $\gamma = 0.03$. As we see, our sparsified K-means algorithm returns fairly accurate estimates of the true cluster centers in one pass over the data, which represent the three classes of digits in the given unlabeled dataset. However, as described, feature-based algorithms require one more pass over the full dataset after finding assignments to return meaningful estimates of the true cluster centers.

Why does our method give effective 1-pass center estimates, while the other methods do not, even if they have comparable accuracy in estimating assignments? The reason is that each sample $\mathbf{x}_i$ is sampled with an *independent copy* of the random sampling operator $\mathbf{R}_i \mathbf{R}_i^T$, and this leads to a *consistent* estimator. For simplicity, assume assignments have been made correctly for a given cluster k, so we know $\mathcal{I}_k$. Then Thm. 7 bounds $\|\mathbf{H}_k - \mathbf{I}_p\|_2$ in terms of $n_k := |\mathcal{I}_k|$, and in particular, we know $\mathbf{H}_k$ converges to $\mathbf{I}_p$ almost surely as $n_k \to \infty$ (this follows from the strong law of large numbers). To be specific, recall that $\boldsymbol{\mu}_k = \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} \mathbf{x}_i$ (assume $\mathbf{x}_i$ are deterministic, though the argument adapts to stochastic $\mathbf{x}_i$ under mild assumptions such as
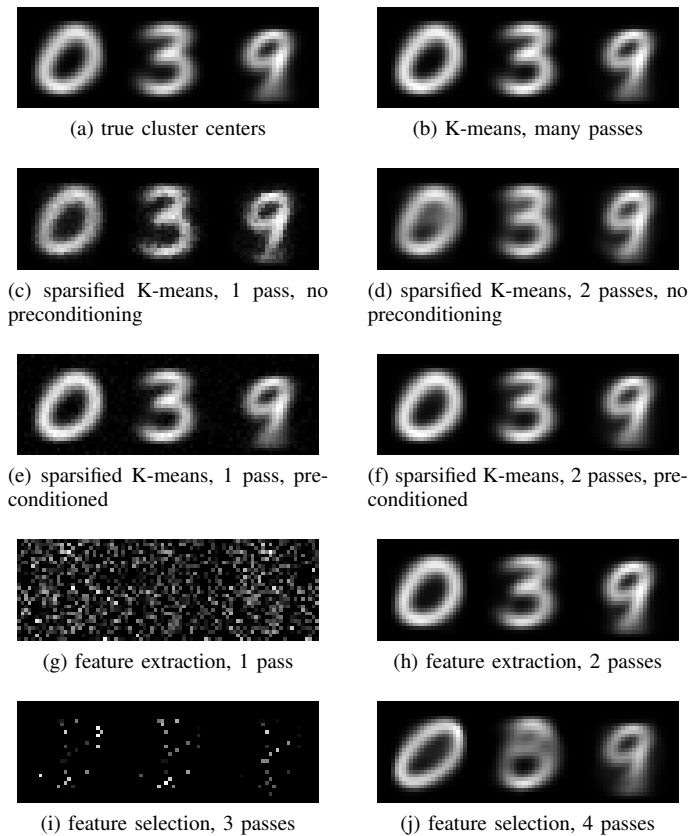
(a) true cluster centers

(b) K-means, many passes

(c) sparsified K-means, 1 pass, no preconditioning

(d) sparsified K-means, 2 passes, no preconditioning

(e) sparsified K-means, 1 pass, pre-conditioned

(f) sparsified K-means, 2 passes, pre-conditioned

(g) feature extraction, 1 pass

(h) feature extraction, 2 passes

(i) feature selection, 3 passes

(j) feature selection, 4 passes

Figure 9. Center estimates $\boldsymbol{\mu}$ from low-pass K-means algorithms.

finite first two moments), then from the sampled data we can form the center estimate

$$\widehat{\boldsymbol{\mu}_k} = \frac{1}{n_k} \frac{p}{m} \sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i$$

and $\widehat{\boldsymbol{\mu}_k} \to \boldsymbol{\mu}_k$ almost surely as $n_k \to \infty$, which follows from the strong law of large numbers and the independence of the $\mathbf{R}_i$.

For feature extraction (FE), the collected data are $\{\boldsymbol{\Omega}\mathbf{x}_i\}_{i \in \mathcal{I}_k}$, so the obvious center estimate is

$$\widehat{\boldsymbol{\mu}_k}^{\text{FE}} = \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} \boldsymbol{\Omega}^\dagger \boldsymbol{\Omega} \mathbf{x}_i = \frac{1}{n_k} \boldsymbol{\Omega}^\dagger \boldsymbol{\Omega} \sum_{i \in \mathcal{I}_k} \mathbf{x}_i$$

with † representing the pseudo-inverse. As $n_k \to \infty$, this does *not* converge to $\boldsymbol{\mu}_k$ because $\boldsymbol{\Omega}^\dagger \boldsymbol{\Omega} \neq \mathbf{I}_p$ (equality is impossible because the term on the left has rank $m < p$). That is, even with more data, the center estimate does not improve because a single copy of the random variable $\boldsymbol{\Omega}$ is used to compress all the data, so it is not consistent. The only solution is to take another pass through the original data $\{\mathbf{x}_i\}$ using the estimated cluster assignments to form the sample center estimate.

C. Big data tests

We further test on two increasingly large extensions of MNIST. On the largest extension, we test an out-of-core memory version of our algorithm. Because of the size of the data, we no longer run the K-means algorithm on all the data in order to provided a benchmark, but since the data are generated similarly to the classic MNIST, it would be reasonable to expect that basic K-means would behave similarly and achieve an accuracy close to 92% as in Fig. 7.

Since the algorithms take longer to run, we reduce the number of replicates in each trial to 10, and perform only 10 trials per algorithm. We focus on feature extraction and not feature selection since the smaller-scale tests indicated feature extraction performed better in both speed, accuracy and number of passes through the data.

c) In-core memory with $n = 6 \cdot 10^5$: Clustering was performed on data from the first 200,000 samples each of the "0", "3" and "9" digits from the mnist8M dataset [48], using code from the "Infinite MNIST" project[6], thus $n = 6 \cdot 10^5$, with
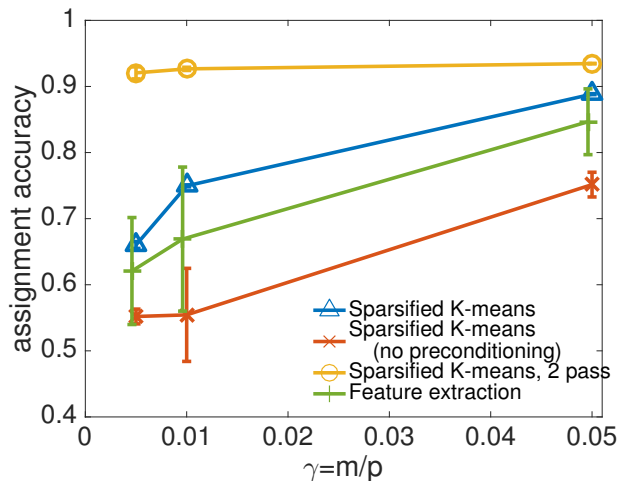
[6]http://leon.bottou.org/projects/infimnist

Figure 10. Similar to Fig. 7 but on much larger data, $n = 6 \cdot 10^5$. The preconditioning helps significantly to increase accuracy, as well as lower variance.

Table III

FROM THE SAME $n = 6 \cdot 10^5$ SIMULATIONS AS FIG. 10, AT $\gamma = 0.05$. ALL NUMBERS ARE AVERAGES OVER THE 10 TRIALS, TIMES IN SECONDS. THE K-MEANS ALGORITHM WAS LIMITED TO 100 ITERATIONS, AND AN ASTERISK DENOTES THE ALGORITHM NEVER CONVERGED WITHIN THIS LIMIT (ON ALL TRIALS).

Algorithm	Total time	Time to sample	Time to precondition	Iterations of K-means
Sparsified K-means	228.8 s	6.0 s	33.7 s	42.1
Sparsified K-means, 2 pass	237.0 s	6.0 s	33.7 s	42.1
Sparsified K-means, no preconditioning	665.6 s	5.1 s	NA	100.0*
Feature extraction	123.1 s	0.4 s	NA	41.9

$p = 784$ as before. This new dataset artificially creates more training examples by applying pseudo-random deformations and translations to the MNIST images.

Accuracy results are in Fig. 10. The preconditioned version of sparsified K-means is much more accurate than the non-preconditioned version, and has better accuracy than feature extraction while also enjoying lower variance and taking only a single pass through the data. If we compute a second pass through the data, accuracy jumps to nearly optimal levels as soon as we sample at least 1% of the data.

Timing results for $\gamma = 0.05$ are in Table III. Feature extraction reduces dimension instead of increasing sparsity, and while both algorithms take roughly the same number of flops, feature extraction has roughly a $2\times$ edge in speed since it has simpler data structures which have better data locality and can be exploited by many algorithms. The timing results are broken down into fine detail to show that the majority of time is in the actual K-means iteration on the reduced/sparsified data. We also note that without preconditioning, K-means never converges within 100 iterations in our tests, a sign that it does not capture the structure of the data, and this greatly contributes to the runtime of the algorithm.

d) Out-of-core memory with $n = 10^7$: We implement out-of-core versions of the sparsified K-means algorithm and the feature extraction algorithm, which efficiently load and compress the data in a batched manner such that the entire matrix is never loaded all at once into the main memory of the computer. The dataset is created using the same "Infinite MNIST" code, so the setup is the same as the previous sections, i.e., $p = 784$, but now $n = 9,631,605$, having 3,168,805, 3,280,085 and 3,182,715 examples of the digits "0", "3" and "9", respectively. Stored in double-precision using Matlab's default compression, which automatically reduces precision if possible, the matrix is 4.9 GB, and would be 56 GB if loaded into memory in double-precision. When loading, the matrix is split into 58 chunks, each one (except the last) with dimensions 784×167183 and approximately 1 GB in size. We repeat the experiment three times, using $\gamma \in \{0.01, 0.05\}$, and 10 replicates.

Table IV shows the results. Accuracy is similar to the $n = 600,000$ simulation, which is not surprising since the data for both experiments were algorithmically generated from the Infinite MNIST code. As expected, time to load data from disk was significant. For example, in the second pass over data at $\gamma = .05$, loading the data required 125 seconds while the actual time for computing for the updated mean and assignment, i.e. Alg. 2 except line 2, was just 28 seconds. However, the time to load the data from disk is still not a bottleneck in the overall computation time, since we only need to perform this once (or twice). In a distributed data setting, where loading the data is even more costly but we would also be able to take γ very small, one may prefer the 1-pass variant over the 2-pass variant.

The time to sample is non-negligible since it requires many calls to a pseudo-random number generator and creating sparse arrays. Preconditioning also takes 170 seconds using the DCT, though we remark that the DCT in Matlab is not well-implemented, and Matlab performs an FFT on the same data in under 1 minute since it directly calls the fftw library. Directly

Table IV
FROM THE $n = 9{,}631{,}605$ SIMULATION. ALL NUMBERS ARE AVERAGES OVER THE 3 TRIALS, TIMES IN SECONDS. ACCURACY LISTED AS MEAN $\pm$ STANDARD DEVIATION.

	Algorithm	Accuracy	Iterations of K-means	Time			
				Total	to sample	to precondition	to load data from disk
$\gamma = .01$	Sparsified K-means	$0.745 \pm .0008$	100*	2630s	38s	170s	125s
	Sparsified K-means, 2 pass	$0.927 \pm .0018$	100*	2783s	38s	170s	250s
	Feature extraction	$0.680 \pm .0610$	73.8	1123s	18s	NA	128s
$\gamma = .05$	Sparsified K-means	$0.887 \pm .0002$	53.5	4380s	137s	159s	126s
	Sparsified K-means, 2 pass	$0.933 \pm .0001$	53.5	4538s	137s	159s	254s
	Feature extraction	$0.836 \pm .0714$	70.4	3384s	62s	NA	129s

Table V
ESTIMATED SPEEDUP. FROM THE $n = 9{,}631{,}605$ SIMULATION, AT $\gamma = 0.05$.

Algorithm	Time to find assignments		Time to update all centers		Combined time	
	Absolute	Speedup	Absolute	Speedup	Absolute	Speedup
K-means	130.0s	1×	150.8s	1×	280.8s	1×
Sparsified K-means	1.3s	100×	5.7s	26.4×	7.0s	40.1×

calling fftw's DCT routine would likely be much faster; it would also be possible to accelerate this step using general-purpose graphical processing units (GP-GPU) since applying the DCT to an array is efficiently parallelized.

For a general idea of how long K-means would take without sub-sampling, we run a single iteration of K-means and compare the results in Table V, finding that we reduce the computational time by a factor of almost 40. The actual time for the full K-means algorithm would not improve by quite this much, since it is not clear if the number of iterations would change, and there is also the fixed cost of loading the data once (for sparsified K-means) or twice (for the 2-pass variant). We also caution that this was a single run of a single iteration of K-means, so the factor 40 is of limited precision, but overall, this is better than the result we hope to see since we have $\gamma = 0.05$ meaning we have kept $1/20^{\text{th}}$ of the data. One reason for seeing 40 times speedup instead of 20 times speedup may be that Matlab paged memory onto secondary storage for the full version of the algorithm, or that the sparsified data was small enough to fit entirely inside the CPU cache memory (20 MB) rather than RAM.

D. Discussion

On the MNIST data, all the fast algorithms show great speedup over standard K-means, and tunable accuracies that can reach the accuracy of standard K-means as $\gamma \to 1$. In our tests, the one-pass (preconditioned) sparsified K-means algorithm appears to be slightly more accurate than feature extraction, and significantly more accurate than feature selection and the non-preconditioned sparsified K-means. In addition, our sparsified K-means has significantly lower variance than feature extraction, which means that the output of our method is more reliable and closer to the output of K-means on the full data among different initializations.

Furthermore, based on the MNIST experiments, if one can afford two passes over the data, the accuracy of our two-pass sparsified K-means reaches the accuracy of standard K-means and, at the same time, accurately estimates the cluster centers. For in-core problems, there is negligible cost for the extra pass, so the two-pass variant is the best candidate. For out-of-core problems, the one-pass variant may be preferred.

VIII. CONCLUSIONS

We have presented a compression scheme for large-scale data sets which leads to both computational time and memory benefits in unsupervised learning tasks such as PCA and K-means clustering. A main feature of our approach is that it requires just one pass over the data thanks to the randomized preconditioning transformation, which makes it applicable to streaming and distributed data settings. In fact, the preconditioning transformation is an essential component of our approach which allows us to achieve accurate and reliable estimates in the data sparsification process and eliminates the need to revisit past entries of the data. A side-benefit of the preconditioning is a reduction in the variance of estimates.

Our sparsified K-means algorithm returns both assignments and cluster centers in a single pass over the data, whereas the state-of-the-art feature-based algorithms require at least two passes. Moreover, our approach leads to per-step guarantees on the clustering structure, as opposed to the guarantees on the overall objective function in feature-based algorithms.

Finally, our compression scheme has the potential to be applicable in many other techniques in signal processing and machine learning, such as subspace learning, K-nearest neighbors, soft K-means, mixture models, and expectation-maximization algorithms. In these settings, the preconditioning and sampling technique could be used to either speed up computation for in-core memory problems, or to create one-pass variants for out-of-core or streaming problems.

APPENDIX A
EXPONENTIAL CONCENTRATION INEQUALITIES

Below are standard inequalities listed in expedient formats.

Theorem A1 (Hoeffding's Inequality [49]). *Let $z_1, \ldots, z_n$ be independent, centered random variables, and assume that each one is bounded:*

$$\mathbb{E}\left[z_k\right] = 0 \text{ and } |z_k| \leq L_k \text{ for each } k = 1, \ldots, n.$$

Introduce the sum $S = \sum_{k=1}^n z_k$, and let $\sigma^2 = \sum_{k=1}^n L_k^2$. Then, for all $t \geq 0$:

$$\mathbb{P}\left\{|S| \geq t\right\} \leq 2 \exp\left(\frac{-t^2/2}{\sigma^2}\right).$$

Theorem A2 (Bernstein Inequality [49]). *Let $z_1, \ldots, z_n$ be independent, centered random variables, and assume that each one is uniformly bounded:*

$$\mathbb{E}\left[z_k\right] = 0 \text{ and } |z_k| \leq L \text{ for each } k = 1, \ldots, n.$$

Introduce the sum $S = \sum_{k=1}^n z_k$, and let $\sigma^2 = \sum_{k=1}^n \mathbb{E}[z_k^2]$ denote the variance of the sum. Then, for all $t \geq 0$:

$$\mathbb{P}\left\{|S| \geq t\right\} \leq 2 \exp\left(\frac{-t^2/2}{\sigma^2 + Lt/3}\right).$$

Theorem A3 (Matrix Bernstein Inequality [50]). *Let $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$ be independent, symmetric, centered random matrices with dimension p, and assume that each one is uniformly bounded:*

$$\mathbb{E}\left[\mathbf{Z}_k\right] = \mathbf{0} \text{ and } \|\mathbf{Z}_k\|_2 \leq L \text{ for each } k = 1, \ldots, n.$$

Introduce the sum $\mathbf{S} = \sum_{k=1}^n \mathbf{Z}_k$, and let $\sigma^2 = \left\|\sum_{k=1}^n \mathbb{E}\left[\mathbf{Z}_k^2\right]\right\|_2$ denote the variance. Then, for all $t \geq 0$:

$$\mathbb{P}\left\{\|\mathbf{S}\|_2 \geq t\right\} \leq p \exp\left(\frac{-t^2/2}{\sigma^2 + Lt/3}\right).$$

APPENDIX B
PROPERTIES OF THE SAMPLING MATRIX

Theorem B4. *Consider a sampling matrix $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_m] \in \mathbb{R}^{p \times m}$, where the m columns are chosen uniformly at random from the set of all p canonical basis vectors without replacement. Then, these columns form an orthonormal basis, i.e., $\mathbf{R}^T \mathbf{R} = \mathbf{I}_m$. Moreover, we have:*

$$\mathbb{E}[\mathbf{R}\mathbf{R}^T] = \frac{m}{p}\mathbf{I}_p \tag{49}$$

and for any fixed vector $\mathbf{x} \in \mathbb{R}^p$ and $m \geq 2$:

$$\mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T] = \frac{m(m-1)}{p(p-1)}\mathbf{x}\mathbf{x}^T + \frac{m(p-m)}{p(p-1)}\operatorname{diag}(\mathbf{x}\mathbf{x}^T). \tag{50}$$

Proof: The columns of $\mathbf{R}$ are distinct canonical basis vectors, thus $\mathbf{R}^T\mathbf{R} = \mathbf{I}_m$. To prove (49), note that $\mathbb{E}[\mathbf{R}\mathbf{R}^T] = \sum_{i=1}^m \mathbb{E}[\mathbf{r}_i\mathbf{r}_i^T]$, and we will show that

$$\mathbb{E}[\mathbf{r}_i\mathbf{r}_i^T] = \frac{1}{p}\mathbf{I}_p \text{ for } i = 1, \ldots, m. \tag{51}$$

The main difficulty is that the columns are dependent on each other since the sampling is without replacement. Let us first consider $i = 1$. In this case, the first column $\mathbf{r}_1$ is chosen uniformly at random from the set of all canonical basis vectors, i.e., $\mathbb{P}\left\{\mathbf{r}_1 = \mathbf{e}_{r_1}\right\} = \frac{1}{p}$ for $r_1 = 1, \ldots, p$. Thus

$$\mathbb{E}\left[\mathbf{r}_1\mathbf{r}_1^T\right] = \sum_{r_1=1}^p \mathbb{P}\left\{\mathbf{r}_1 = \mathbf{e}_{r_1}\right\}\mathbf{e}_{r_1}\mathbf{e}_{r_1}^T = \frac{1}{p}\sum_{r_1=1}^p \mathbf{e}_{r_1}\mathbf{e}_{r_1}^T = \frac{1}{p}\mathbf{I}_p.$$

For $i \in \{2, \ldots, m\}$, we compute the expectation as follows:

$$
\begin{aligned}
\mathbb{E}\left[\mathbf{r}_i \mathbf{r}_i^T\right] &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{r}_i \mathbf{r}_i^T | \mathbf{r}_1, \ldots, \mathbf{r}_{i-1}\right]\right] \\
&= \sum_{(r_1, \ldots, r_{i-1})} \mathbb{P}\left\{\mathbf{r}_1 = \mathbf{e}_{r_1}, \ldots \mathbf{r}_{i-1} = \mathbf{e}_{r_{i-1}}\right\} \\
&\quad \times \mathbb{E}\left[\mathbf{r}_i \mathbf{r}_i^T | \mathbf{r}_1 = \mathbf{e}_{r_1}, \ldots \mathbf{r}_{i-1} = \mathbf{e}_{r_{i-1}}\right]
\end{aligned}
$$

where the summation is over the set of $(i-1)$ distinct values from $\{1, \ldots, p\}$, thus $\mathbb{P}\{\mathbf{r}_1 = \mathbf{e}_{r_1}, \ldots, \mathbf{r}_{i-1} = \mathbf{e}_{r_{i-1}}\} = \frac{1}{p}\frac{1}{p-1} \cdots \frac{1}{p-(i-2)}$. Also, the expectation does not depend on the permutation of $r_1, \ldots, r_{i-1}$, so we condense the sum to range over just the set, not permutation, of distinct values, and adjust by multiplying by $(i-1)!$. Therefore,

$$
\begin{aligned}
\mathbb{E}\left[\mathbf{r}_i \mathbf{r}_i^T\right] &= (i-1)! \left(\frac{1}{p}\frac{1}{p-1} \cdots \frac{1}{p-(i-2)}\right) \\
&\quad \times \sum_{\substack{\{r_1, \ldots, r_{i-1}\} \\ \text{distinct}}} \left(\sum_{\substack{r_i \text{ from} \\ \text{remaining values}}} \mathbb{P}\{\mathbf{r}_i = \mathbf{e}_{r_i}\} \mathbf{e}_{r_i} \mathbf{e}_{r_i}^T\right) \\
&\stackrel{(a)}{=} (i-1)! \left(\frac{1}{p}\frac{1}{p-1} \cdots \frac{1}{p-(i-2)}\frac{1}{p-(i-1)}\right) \\
&\quad \times \left(\sum_{\substack{\{r_1, \ldots, r_{i-1}\} \\ \text{distinct}}} \sum_{\substack{r_i \text{ from} \\ \text{remaining values}}} \mathbf{e}_{r_i} \mathbf{e}_{r_i}^T\right) \\
&\stackrel{(b)}{=} (i-1)! \left(\frac{1}{p}\frac{1}{p-1} \cdots \frac{1}{p-(i-2)}\frac{1}{p-(i-1)}\right) \\
&\quad \times \binom{p-1}{i-1}\mathbf{I}_p = \frac{1}{p}\mathbf{I}_p
\end{aligned}
$$

where (a) follows from $\mathbb{P}\{\mathbf{r}_i = \mathbf{e}_{r_i}\} = \frac{1}{p-(i-1)}$ and (b) is obtained by counting the number of cases where $r_i = j$, $1 \leq j \leq p$, and this can be easily computed by counting the number of cases that j is not in the set $\{r_1, \ldots, r_{i-1}\}$ which is $\binom{p-1}{i-1}$. This completes the proof of (49).

Next, we show that (50) holds. Note that:

$$
\begin{aligned}
\mathbb{E}\left[\mathbf{R}\mathbf{R}^T \mathbf{x}\mathbf{x}^T \mathbf{R}\mathbf{R}^T\right] &= \sum_{(r_1, \ldots, r_m)} \mathbb{P}\left\{\mathbf{r}_1 = \mathbf{e}_{r_1}, \ldots \mathbf{r}_m = \mathbf{e}_{r_m}\right\} \\
&\quad \times \left(\sum_{i=1}^m \mathbf{e}_{r_i} \mathbf{e}_{r_i}^T\right) \mathbf{x}\mathbf{x}^T \left(\sum_{i=1}^m \mathbf{e}_{r_i} \mathbf{e}_{r_i}^T\right) \\
&= \left(\frac{1}{p}\frac{1}{p-1} \cdots \frac{1}{p-(m-1)}\right) \left\{\alpha_1 \sum_{k=1}^p \mathbf{e}_k \mathbf{e}_k^T \mathbf{x}\mathbf{x}^T \mathbf{e}_k \mathbf{e}_k^T\right. \\
&\quad \left. + \alpha_2 \sum_{k \neq l} \mathbf{e}_k \mathbf{e}_k^T \mathbf{x}\mathbf{x}^T \mathbf{e}_l \mathbf{e}_l^T\right\}
\end{aligned}
$$

where the summation is over the set of m distinct values from $\{1, \ldots, p\}$ and we should find the coefficients α_1 and α_2. In fact, α_1 represents the number of cases that each k, $1 \leq k \leq p$, is among the m numbers chosen from $\{1, \ldots, p\}$ without replacement. Let's fix $r_1 = 1$, we then have $\binom{p-1}{m-1}(m-1)!$ cases. Thus, we see that:

$$
\alpha_1 = \left[\binom{p-1}{m-1}(m-1)!\right] m = \frac{m(p-1)(p-2)!}{(p-m)!}.
$$

Similarly, α_2 represents the number of cases that each pair of k and l, $1 \leq k, l \leq p$ and $k \neq l$, is among the m numbers chosen from $\{1, \ldots, p\}$. Let's fix $r_1 = 1$ and $r_2 = 2$, we then have $\binom{p-2}{m-2}(m-2)!$ cases. This argument leads to:

$$
\alpha_2 = \left[\binom{p-2}{m-2}(m-2)!\right] m(m-1) = \frac{m(m-1)(p-2)!}{(p-m)!}.
$$

Since $\alpha_2 < \alpha_1$, we can write the expectation as follows:

$$
\begin{aligned}
\mathbb{E}\left[\mathbf{R}\mathbf{R}^T \mathbf{x}\mathbf{x}^T \mathbf{R}\mathbf{R}^T\right] &= \left(\frac{1}{p}\frac{1}{p-1} \cdots \frac{1}{p-(m-1)}\right) \\
&\quad \times \left(\alpha_2 \left(\sum_{k=1}^p \mathbf{e}_k \mathbf{e}_k^T\right) \mathbf{x}\mathbf{x}^T \left(\sum_{k=1}^p \mathbf{e}_k \mathbf{e}_k^T\right) + (\alpha_1 - \alpha_2) \operatorname{diag}\left(\mathbf{x}\mathbf{x}^T\right)\right)
\end{aligned}
$$

and this completes the proof. ∎

Lemma B5. *Suppose we sample m entries of $\mathbf{x} \in \mathbb{R}^p$ uniformly at random without replacement. Then, the probability of keeping the j-th entry of $\mathbf{x}$ is $\frac{m}{p}$ for all $j = 1, \ldots, p$.*

Proof: The proof follows from the properties of sampling matrices given in Theorem B4. For a matrix $\mathbf{R} \in \mathbb{R}^{p \times m}$ containing m distinct canonical basis vectors, we show that $\mathbb{E}[\mathbf{RR}^T] = \frac{m}{p}\mathbf{I}_p$. Each component of the sub-sampled data $\mathbf{w} = \mathbf{RR}^T\mathbf{x}$ is a random variable with two possible values: the j-th entry of $\mathbf{w}$ takes the same value as the corresponding entry of $\mathbf{x}$ with probability π_j and zero otherwise. Since $\mathbb{E}[\mathbf{w}] = \mathbb{E}[\mathbf{RR}^T\mathbf{x}] = \frac{m}{p}\mathbf{x}$, we conclude that $\pi_j = \frac{m}{p}$ for $j = 1, \ldots, p$. ∎

APPENDIX C
PROOF OF THEOREM 6

We present the proof of Thm. 6 on the covariance estimator. First, we show that $\widehat{\mathbf{C}}_n$ is an unbiased estimator, i.e., $\mathbb{E}[\widehat{\mathbf{C}}_n] = \mathbf{C}_{\text{emp}}$. Using Thm. B4, we compute the following expectations:

$$\mathbb{E}[\widehat{\mathbf{C}}_{\text{emp}}] = \mathbf{C}_{\text{emp}} + \frac{(p-m)}{(m-1)}\,\text{diag}(\mathbf{C}_{\text{emp}}) \tag{52}$$

and

$$\mathbb{E}[\text{diag}(\widehat{\mathbf{C}}_{\text{emp}})] = \text{diag}(\mathbb{E}[\widehat{\mathbf{C}}_{\text{emp}}]) = \frac{(p-1)}{(m-1)}\,\text{diag}(\mathbf{C}_{\text{emp}}). \tag{53}$$

Hence, using (52) and (53), we get $\mathbb{E}[\widehat{\mathbf{C}}_n] = \mathbf{C}_{\text{emp}}$. To find the closeness of $\widehat{\mathbf{C}}_n$ to its expectation $\mathbf{C}_{\text{emp}}$, we use the matrix Bernstein inequality (Thm. A3). Note that $(\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}})$ can be written as a sum of n independent centered random matrices:

$$\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}} = \sum_{i=1}^n \mathbf{Z}_i = \sum_{i=1}^n \frac{1}{n}(\mathbf{Z}_i^{(1)} - \mathbf{Z}_i^{(2)} - \mathbf{Z}_i^{(3)}) \tag{54}$$

where $\mathbf{Z}_i^{(1)} = \frac{p(p-1)}{m(m-1)}\mathbf{w}_i\mathbf{w}_i^T$, $\mathbf{Z}_i^{(2)} = \frac{p(p-m)}{m(m-1)}\text{diag}(\mathbf{w}_i\mathbf{w}_i^T)$, $\mathbf{Z}_i^{(3)} = \mathbf{x}_i\mathbf{x}_i^T$, and $\mathbf{w}_i = \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i$ is the sub-sampled data. To apply matrix Bernstein, we should find a uniform bound on the spectral norm of each summand $\|\mathbf{Z}_i\|_2$. We find a uniform bound for the spectral norm of $\mathbf{Z}_i^{(1)}$:

$$\|\mathbf{Z}_i^{(1)}\|_2 = \frac{p(p-1)}{m(m-1)}\|\mathbf{w}_i\mathbf{w}_i^T\|_2 = \frac{p(p-1)}{m(m-1)}\|\mathbf{w}_i\|_2^2$$
$$\leq \frac{p(p-1)}{m(m-1)}\rho\|\mathbf{x}_i\|_2^2 \leq \frac{p(p-1)}{m(m-1)}\rho\|\mathbf{X}\|_{\text{max-col}}^2. \tag{55}$$

For the second term $\mathbf{Z}_i^{(2)}$, it is easy to verify that $\text{diag}(\mathbf{w}_i\mathbf{w}_i^T) \preccurlyeq \text{diag}(\mathbf{x}_i\mathbf{x}_i^T)$, where $\mathbf{A} \preccurlyeq \mathbf{B}$ means that $\mathbf{B} - \mathbf{A}$ is positive semidefinite, and thus we get $\|\text{diag}(\mathbf{w}_i\mathbf{w}_i^T)\|_2 \leq \|\text{diag}(\mathbf{x}_i\mathbf{x}_i^T)\|_2$ which can be used to bound $\|\mathbf{Z}_i^{(2)}\|_2$:

$$\|\mathbf{Z}_i^{(2)}\|_2 = \frac{p(p-m)}{m(m-1)}\|\text{diag}(\mathbf{w}_i\mathbf{w}_i^T)\|_2 \leq \frac{p(p-m)}{m(m-1)}\|\mathbf{X}\|_{\text{max}}^2. \tag{56}$$

We also find a uniform bound for the spectral norm of $\mathbf{Z}_i^{(3)}$:

$$\|\mathbf{Z}_i^{(3)}\|_2 = \|\mathbf{x}_i\mathbf{x}_i^T\|_2 = \|\mathbf{x}_i\|_2^2 \leq \|\mathbf{X}\|_{\text{max-col}}^2 \tag{57}$$

and the triangle inequality for the spectral norm leads to:

$$\|\mathbf{Z}_i\|_2 \leq \frac{1}{n}\left(\|\mathbf{Z}_i^{(1)}\|_2 + \|\mathbf{Z}_i^{(2)}\|_2 + \|\mathbf{Z}_i^{(3)}\|_2\right) \leq L \tag{58}$$

where L is given in (25).

Next, we compute the variance of our estimator σ^2:

$$\sigma^2 = \|\mathbb{E}[(\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}})^2]\|_2 = \|\sum_{i=1}^n \mathbb{E}[\mathbf{Z}_i^2]\|_2. \tag{59}$$

We find the variance σ^2 by first computing the expectation:

$$\mathbb{E}[\mathbf{Z}_i^2] = \frac{1}{n^2}\mathbb{E}[(\mathbf{Z}_i^{(1)} - \mathbf{Z}_i^{(2)} - \mathbf{Z}_i^{(3)})^2] \tag{60}$$

which requires computing the expectation of the product of terms $\mathbf{Z}_i^{(1)}$, $\mathbf{Z}_i^{(2)}$, and $\mathbf{Z}_i^{(3)}$. We begin by computing:

$$\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(1)}] = \frac{p^2(p-1)^2}{m^2(m-1)^2}\mathbb{E}[\mathbf{w}_i\mathbf{w}_i^T\mathbf{w}_i\mathbf{w}_i^T]$$

$$= \frac{p^2(p-1)^2}{m^2(m-1)^2}\mathbb{E}[\|\mathbf{w}_i\|_2^2\mathbf{w}_i\mathbf{w}_i^T] \preccurlyeq \frac{p^2(p-1)^2}{m^2(m-1)^2}\rho\|\mathbf{x}_i\|_2^2\mathbb{E}[\mathbf{w}_i\mathbf{w}_i^T]$$

$$= \frac{p(p-1)}{m(m-1)}\rho\|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T + \frac{p(p-1)(p-m)}{m(m-1)^2}\rho\|\mathbf{x}_i\|_2^2\,\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T)$$

where the inequality follows the fact that expectation preserves the semidefinite order and we also used Thm. B4 to compute $\mathbb{E}[\mathbf{w}_i\mathbf{w}_i^T]$. Now, we compute $\mathbb{E}[\mathbf{Z}_i^{(2)}\mathbf{Z}_i^{(2)}]$:

$$\mathbb{E}[\mathbf{Z}_i^{(2)}\mathbf{Z}_i^{(2)}] = \frac{p^2(p-m)^2}{m^2(m-1)^2}\mathbb{E}[(\mathrm{diag}(\mathbf{w}_i\mathbf{w}_i^T))^2]$$

$$= \frac{p^2(p-m)^2}{m^2(m-1)^2}\frac{m}{p}(\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T))^2 = \frac{p(p-m)^2}{m(m-1)^2}(\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T))^2$$

where this follows from $\mathbb{E}[(\mathrm{diag}(\mathbf{w}_i\mathbf{w}_i^T))^2] = \frac{m}{p}(\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T))^2$. To see this, let $w_{i,j}$ denote the j-th element of $\mathbf{w}_i$ and note that $\mathrm{diag}(\mathbf{w}_i\mathbf{w}_i^T)$ is a diagonal matrix where the j-th element is $w_{i,j}^2$. Thus, $(\mathrm{diag}(\mathbf{w}_i\mathbf{w}_i^T))^2$ is also a diagonal matrix where the j-th element is equal to $w_{i,j}^4$. Based on Lemma B5, the probability of keeping the j-th element of $\mathbf{x}_i$ under the uniform sampling without replacement is $\frac{m}{p}$, which means that $\mathbb{E}[w_{i,j}^4] = \frac{m}{p}x_{i,j}^4$.

Next, we can easily find the following expectations:

$$\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(3)}] = \mathbb{E}[\mathbf{Z}_i^{(3)}\mathbf{Z}_i^{(1)}]^T = \frac{p(p-1)}{m(m-1)}\mathbb{E}[\mathbf{w}_i\mathbf{w}_i^T]\mathbf{x}_i\mathbf{x}_i^T$$

$$= \|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T + \frac{(p-m)}{(m-1)}\,\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T)\mathbf{x}_i\mathbf{x}_i^T$$

and

$$\mathbb{E}[\mathbf{Z}_i^{(2)}\mathbf{Z}_i^{(3)}] = \mathbb{E}[\mathbf{Z}_i^{(3)}\mathbf{Z}_i^{(2)}]^T = \frac{p(p-m)}{m(m-1)}\mathbb{E}[\mathrm{diag}(\mathbf{w}_i\mathbf{w}_i^T)]\mathbf{x}_i\mathbf{x}_i^T$$

$$= \frac{(p-m)}{(m-1)}\,\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T)\mathbf{x}_i\mathbf{x}_i^T$$

and

$$\mathbb{E}[\mathbf{Z}_i^{(3)}\mathbf{Z}_i^{(3)}] = \mathbf{x}_i\mathbf{x}_i^T\mathbf{x}_i\mathbf{x}_i^T = \|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T.$$

Hence, based on the expectations computed above and the triangle inequality, we get:

$$\sigma^2 = \left\|\sum_{i=1}^n \mathbb{E}[\mathbf{Z}_i^2]\right\|_2 \leq \frac{1}{n}\left(\frac{p(p-1)}{m(m-1)}\rho - 1\right)\cdot\left\|\frac{1}{n}\sum_{i=1}^n\|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T\right\|_2$$

$$+ \frac{1}{n}\frac{p(p-1)(p-m)}{m(m-1)^2}\rho\left\|\frac{1}{n}\sum_{i=1}^n\|\mathbf{x}_i\|_2^2\,\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T)\right\|_2$$

$$+ \frac{1}{n^2}\frac{p(p-m)^2}{m(m-1)^2}\left\|\sum_{i=1}^n(\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T))^2\right\|_2$$

$$+ \frac{1}{n^2}\left(\left\|\sum_{i=1}^n\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}]\right\|_2 + \left\|\sum_{i=1}^n\mathbb{E}[\mathbf{Z}_i^{(2)}\mathbf{Z}_i^{(1)}]\right\|_2\right).$$

We also have the following two inequalities:

$$\frac{1}{n}\sum_{i=1}^n\|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T \preccurlyeq \|\mathbf{X}\|_{\mathrm{max-col}}^2\cdot\mathbf{C}_{\mathrm{emp}}$$

and

$$\frac{1}{n}\sum_{i=1}^n\|\mathbf{x}_i\|_2^2\,\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T) \preccurlyeq \|\mathbf{X}\|_{\mathrm{max-col}}^2\cdot\mathrm{diag}(\mathbf{C}_{\mathrm{emp}}).$$

In the last step, we find an upper bound for the following:

$$\left\|\sum_{i=1}^{n}\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}]\right\|_2 \le \sum_{i=1}^{n}\|\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}]\|_2$$

$$\le \sum_{i=1}^{n}\mathbb{E}[\|\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}\|_2] \le \sum_{i=1}^{n}\mathbb{E}[\|\mathbf{Z}_i^{(1)}\|_2\|\mathbf{Z}_i^{(2)}\|_2]$$

where this follows from the triangle inequality, Jensen's inequality, and the fact that for two symmetric matrices $\mathbf{A}$ and $\mathbf{B}$, we have $\|\mathbf{AB}\|_2 \le \|\mathbf{A}\|_2\|\mathbf{B}\|_2$. We compute the two terms inside the expectation:

$$\|\mathbf{Z}_i^{(1)}\|_2 = \frac{p(p-1)}{m(m-1)}\|\mathbf{w}_i\|_2^2 = \frac{p(p-1)}{m(m-1)}\mathbf{x}_i^T\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i$$

and

$$\|\mathbf{Z}_i^{(2)}\|_2 = \frac{p(p-m)}{m(m-1)}\|\operatorname{diag}(\mathbf{w}_i\mathbf{w}_i^T)\|_2 \le \frac{p(p-m)}{m(m-1)}\|\mathbf{X}\|_{\max}^2.$$

Hence, using the property $\mathbb{E}[\mathbf{R}_i\mathbf{R}_i^T] = \frac{m}{p}\mathbf{I}_p$, we get:

$$\mathbb{E}[\|\mathbf{Z}_i^{(1)}\|_2\|\mathbf{Z}_i^{(2)}\|_2] \le \frac{p^2(p-1)(p-m)}{m^2(m-1)^2}\|\mathbf{X}\|_{\max}^2\mathbb{E}[\mathbf{x}_i^T\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i]$$

$$= \frac{p(p-1)(p-m)}{m(m-1)^2}\|\mathbf{X}\|_{\max}^2\|\mathbf{x}_i\|_2^2$$

and using $\|\mathbf{X}\|_F^2 = \sum_{i=1}^{n}\|\mathbf{x}_i\|_2^2$, we have:

$$\left\|\sum_{i=1}^{n}\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}]\right\|_2 \le \frac{p(p-1)(p-m)}{m(m-1)^2}\|\mathbf{X}\|_{\max}^2\|\mathbf{X}\|_F^2$$

and this completes the proof.

APPENDIX D
PRESERVATION OF PAIRWISE DISTANCES

Theorem D6. *Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two fixed vectors in $\mathbb{R}^p$. Consider the structured dimension reduction map consisting of the preconditioning transformation $\mathbf{HD}$ (1) and the sampling matrix $\mathbf{R} \in \mathbb{R}^{p\times m}$, where the m columns are chosen uniformly at random from the set of all canonical basis vectors without replacement. Then, with probability at least $1 - 3\beta^{-1}$,*

$$0.40\,\|\mathbf{x}_1 - \mathbf{x}_2\|_2 \le \left\|\sqrt{\frac{p}{m}}\mathbf{R}^T\mathbf{HD}(\mathbf{x}_1 - \mathbf{x}_2)\right\|_2 \le 1.48\,\|\mathbf{x}_1 - \mathbf{x}_2\|_2 \tag{61}$$

given that $4\left[\sqrt{\beta} + \sqrt{8\log(\beta p)}\right]^2\log(\beta) \le m \le p$.

Proof: This result is a straightforward consequence of Theorem 3.1 in [43]. Let us denote $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$ and represent it as $\mathbf{x} = \mathbf{Vc}$, where $\mathbf{V} \in \mathbb{R}^{p\times\beta}$, $\beta < m$, is an orthonormal matrix and $\mathbf{c} \in \mathbb{R}^{\beta}$ (e.g. the first column of $\mathbf{V}$ is $\mathbf{x}/\|\mathbf{x}\|_2$ and the remaining $(m-1)$ columns can be chosen via Gram-Schmidt). We then have the following deterministic lower and upper bounds for $\|\mathbf{R}^T\mathbf{HDx}\|_2 = \|\mathbf{R}^T\mathbf{HDVc}\|_2$:

$$\sigma_\beta(\mathbf{R}^T\mathbf{HDV})\|\mathbf{c}\|_2 \le \|\mathbf{R}^T\mathbf{HDVc}\|_2 \le \sigma_1(\mathbf{R}^T\mathbf{HDV})\|\mathbf{c}\|_2$$

where σ_1 and σ_β denote the largest and smallest singular values. Based on [43], for $m \ge 4[\sqrt{\beta} + \sqrt{8\log(\beta p)}]^2\log(\beta)$ and with probability at least $1 - 3\beta^{-1}$,

$$0.40\sqrt{m/p} \le \sigma_\beta(\mathbf{R}^T\mathbf{HDV}), \quad \sigma_1(\mathbf{R}^T\mathbf{HDV}) \le 1.48\sqrt{m/p}.$$

Note that $\|\mathbf{c}\|_2 = \|\mathbf{Vc}\|_2 = \|\mathbf{x}\|_2$ since $\mathbf{V}$ is an orthonormal matrix and this completes the proof. ∎

REFERENCES

[1] I. Jolliffe, *Principal component analysis*, 2nd ed. Springer-Verlag New York, 2002.

[2] S. Muthukrishnan, *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[3] K. Slavakis, G. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics: (statistical) learning tools for our era of data deluge," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 18–31, 2014.

[4] N. Ailon and B. Chazelle, "The fast Johnson-Lindenstrauss transform and approximate nearest neighbors," *SIAM Journal on Computing*, vol. 39, pp. 302–322, 2009.

[5] T. Tao, *Topics in random matrix theory*, ser. Graduate Studies in Mathematics. American Mathematical Society, 2012, vol. 132.

[6] 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.

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

[8] M. Mahoney, "Randomized algorithms for matrices and data," *Foundations and Trends in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

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

[10] E. Candès and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[11] I. Mitliagkas, C. Caramanis, and P. Jain, "Memory limited, Streaming PCA," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 2886–2894.

[12] J. A. Tropp, "An introduction to matrix concentration inequalities," *Foundations and Trends in Machine Learning*, vol. 8, no. 1-2, pp. 1–230, 2015. [Online]. Available: http://dx.doi.org/10.1561/2200000048

[13] M. Davenport, P. Boufounos, M. Wakin, and R. Baraniuk, "Signal processing with compressive measurements," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, pp. 445–460, 2010.

[14] A. Gilbert, J. Park, and M. Wakin, "Sketched SVD: Recovering spectral features from compressive measurements," *arXiv preprint arXiv:1211.0361*, 2012.

[15] J. Park, M. Wakin, and A. Gilbert, "Modal analysis with compressive measurements," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1655–1670, 2014.

[16] S. Gleichman and Y. Eldar, "Blind compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6958–6975, 2011.

[17] C. Studer and R. Baraniuk, "Dictionary learning from sparsely corrupted or compressed signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 3341–3344.

[18] F. Pourkamali-Anaraki and S. Hughes, "Compressive K-SVD," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 5469–5473.

[19] F. Pourkamali-Anaraki, S. Becker, and S. Hughes, "Efficient dictionary learning via very sparse random projections," in *Sampling Theory and Applications (SampTA)*, 2015, pp. 478–482.

[20] M. Aghagolzadeh and H. Radha, "New guarantees for blind compressed sensing," in *53rd Annual Allerton Conf. on Comm., Control and Computing*, U. Illinois Urbana-Champaign, 2015.

[21] J. Fowler, "Compressive-projection principal component analysis," *IEEE Transactions on Image Processing*, vol. 18, no. 10, pp. 2230–2242, 2009.

[22] H. Qi and S. Hughes, "Invariance of principal components under low-dimensional random projection of the data," in *IEEE International Conference on Image Processing (ICIP)*, 2012, pp. 937–940.

[23] F. Pourkamali-Anaraki and S. Hughes, "Efficient recovery of principal components from compressive measurements with application to Gaussian mixture model estimation," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 2332–2336.

[24] Y. Chen, Y. Chi, and A. Goldsmith, "Exact and stable covariance estimation from quadratic sampling via convex programming," *IEEE Trans. on Information Theory*, vol. 61, pp. 4034–4059, 2015.

[25] F. Pourkamali-Anaraki and S. Hughes, "Memory and computation efficient PCA via very sparse random projections," in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014, pp. 1341–1349.

[26] M. Azizyan, A. Krishnamurthy, and A. Singh, "Extreme compressive sampling for covariance estimation," *arXiv preprint arXiv:1506.00898*, 2015.

[27] P. Li, T. Hastie, and K. Church, "Very sparse random projections," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 287–296.

[28] P. Ma, M. Mahoney, and B. Yu, "A statistical perspective on algorithmic leveraging," in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014, pp. 91–99.

[29] P. Drineas, M. Magdon-Ismail, M. Mahoney, and D. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *The Journal of Machine Learning Research*, pp. 3475–3506, 2012.

[30] M. Ghashami and J. Phillips, "Relative errors for deterministic low-rank matrix approximations," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2014, pp. 707–717.

[31] D. Achlioptas and F. McSherry, "Fast computation of low rank matrix approximations," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, 2001, pp. 611–618.

[32] D. Achlioptas and F. Mcsherry, "Fast computation of low-rank matrix approximations," *J. ACM*, vol. 54, no. 2, Apr. 2007. [Online]. Available: http://doi.acm.org/10.1145/1219092.1219097

[33] D. Achlioptas, Z. Karnin, and E. Liberty, "Near-optimal entrywise sampling for data matrices," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 1565–1573.

[34] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[35] X. Wu and et al., "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[36] C. Boutsidis, A. Zouzias, M. Mahoney, and P. Drineas, "Randomized dimensionality reduction for k-means clustering," *IEEE Transactions on Information Theory*, vol. 61, no. 2, pp. 1045–1062, 2015.

[37] D. Feldman, M. Schmidt, and C. Sohler, "Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering," *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1434–1453, 2013.

[38] M. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, "Dimensionality reduction for k-means clustering and low rank approximation," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, 2015, pp. 163–172.

[39] P. Traganitis, K. Slavakis, and G. Giannakis, "Sketch and validate for big data clustering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 678–690, 2015.

[40] T. Do, L. Gan, N. Nguyen, and T. Tran, "Fast and efficient compressive sensing using structurally random matrices," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 139–154, 2012.

[41] Q. Le, T. Sarlós, and A. Smola, "Fastfood–approximating kernel expansions in loglinear time," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013, pp. 244–252.

[42] V. Sindhwani, T. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 3088–3096.

[43] J. Tropp, "Improved analysis of the subsampled randomized Hadamard transform," *Advances in Adaptive Data Analysis*, pp. 115–126, 2011.

[44] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge University Press, 1991, cambridge Books Online. [Online]. Available: http://dx.doi.org/10.1017/CBO9780511840371

[45] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *SODA*, 2007, pp. 1027–1035.

[46] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near-optimal column-based matrix reconstruction," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 687–717, 2014.

[47] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 881–892, Jul 2002.

[48] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines using selective sampling," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. Cambridge, MA.: MIT Press, 2007, pp. 301–320. [Online]. Available: http://leon.bottou.org/papers/loosli-canu-bottou-2006

[49] P. Massart, *Concentration Inequalities and Model Selection*. Springer-Verlag Berlin Heidelberg, 2003.

[50] L. Mackey, M. Jordan, R. Chen, B. Farrell, and J. Tropp, "Matrix concentration inequalities via the method of exchangeable pairs," *The Annals of Probability*, vol. 42, no. 3, pp. 906–945, 2014.