Randomized Algorithms for Large-Scale Data Analysis

by

Farhad Pourkamali Anaraki

B.S., University of Tehran, 2011

M.S., University of Colorado Boulder, 2014

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Electrical, Computer, and Energy Engineering

2017

This thesis entitled:
Randomized Algorithms for Large-Scale Data Analysis
written by Farhad Pourkamali Anaraki
has been approved for the Department of Electrical, Computer, and Energy Engineering

Prof. Stephen Becker

Prof. Michael B. Wakin

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Pourkamali Anaraki, Farhad (Ph.D., Electrical Engineering)

Randomized Algorithms for Large-Scale Data Analysis

Thesis directed by Prof. Stephen Becker

Massive high-dimensional data sets are ubiquitous in all scientific disciplines. Extracting meaningful information from these data sets will bring future advances in fields of science and engineering. However, the complexity and high-dimensionality of modern data sets pose unique computational and statistical challenges. The computational requirements of analyzing large-scale data exceed the capacity of traditional data analytic tools. The challenges surrounding large high-dimensional data are felt not just in processing power, but also in memory access, storage requirements, and communication costs. For example, modern data sets are often too large to fit into the main memory of a single workstation and thus data points are processed sequentially without a chance to store the full data. Therefore, there is an urgent need for the development of scalable learning tools and efficient optimization algorithms in today's high-dimensional data regimes.

A powerful approach to tackle these challenges is centered around preprocessing high-dimensional data sets via a dimensionality reduction technique that preserves the underlying geometry and structure of the data. This approach stems from the observation that high-dimensional data sets often have intrinsic dimension which is significantly smaller than the ambient dimension. Therefore, information-preserving dimensionality reduction methods are valuable tools for reducing the memory and computational requirements of data analytic tasks on large-scale data sets.

Recently, randomized dimension reduction has received a lot of attention in several fields, including signal processing, machine learning, and numerical linear algebra. These methods use random sampling or random projection to construct low-dimensional representations of the data, known as sketches or compressive measurements. These random-

ized methods are effective in modern data settings since they provide a non-adaptive data-independent mapping of high-dimensional data into a low-dimensional space. However, such methods require strong theoretical guarantees to ensure that the key properties of original data are preserved under a randomized mapping.

This dissertation focuses on the design and analysis of efficient data analytic tasks using randomized dimensionality reduction techniques. Specifically, four efficient signal processing and machine learning algorithms for large high-dimensional data sets are proposed: covariance estimation and principal component analysis, dictionary learning, clustering, and low-rank approximation of positive semidefinite kernel matrices. These techniques are valuable tools to extract important information and patterns from massive data sets. Moreover, an efficient data sparsification framework is introduced that does not require incoherence and distributional assumptions on the data. A main feature of the proposed compression scheme is that it requires only one pass over the data due to the randomized preconditioning transformation, which makes it applicable to streaming and distributed data settings.

The main contribution of this dissertation is threefold: (1) strong theoretical guarantees are provided to ensure that the proposed randomized methods preserve the key properties and structure of high-dimensional data; (2) tradeoffs between accuracy and memory/computation savings are characterized for a large class of data sets as well as dimensionality reduction methods, including random linear maps and random sampling; (3) extensive numerical experiments are presented to demonstrate the performance and benefits of our proposed methods compared to prior works.

Dedication

This dissertation is dedicated to my parents for their love and endless support.

Acknowledgements

I would like to extend my gratitude and appreciation to my advisor, Prof. Stephen Becker, for his support and guidance through my Ph.D. studies. I am honored to have an advisor who is dedicated to research, academics, and providing assistance to his students in order to improve their overall career. Thank you.

I would like to thank Prof. Michael B. Wakin for his support and informative discussions through my Ph.D. studies. His comments helped to improve the overall quality of this dissertation.

In the course of my studies, I have been fortunate to receive funding and travel grants from the National Science Foundation (Grant CCF-1117775), Bloomberg Data Science Research Grant Program, Statistical and Applied Mathematical Sciences Institute (SAMSI), ACM's Special Interest Group on Artificial Intelligence (SIGAI), Institute for Computational and Experimental Research in Mathematics (ICERM), and Norbert Wiener Center for Harmonic Analysis and Applications.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

Due to advances in science and technology over the last decade, researchers have been confronted with an overwhelming amount of complex data. This information stems from social media, imaging devices, sensors, surveillance technology, business and medical platforms. For example, recent advances in neuroimaging techniques, such as fMRI, produce high-resolution brain images which can be used to explore correlations between brain connectivity and diseases [54]. Therefore, extraction of important patterns and structures from modern data have the potential to bring significant advancements to science and technology in order to improve the overall quality of life.

In fact, data is the new currency of today's digital world and learning from large volumes of data will revolutionize many industries. For example, a recent technical report states that, "a one percent reduction in process inefficiency in healthcare would lead to $63 billion in savings over 15 years" [53]. This exemplifies the significance of efficient learning methods and data analytic tasks in the era of the data deluge.

The emerging modern data sets are characterized by their massive sample sizes and high-dimensionality. These two properties come with several challenges that data scientists have to address in order to extract meaning information in an efficient manner. As a result, there are ongoing cross-disciplinary efforts to develop the necessary theoretical foundations to scale learning algorithms to massive high-dimensional data. The following paragraphs describe three major problems for analyzing modern data sets.

Computational complexity The high dimensionality of massive data sets makes the traditional learning and inference methods computationally inefficient. This is mainly due to the fact that the runtime of most data analytic tasks depends at least linearly on the dimension of data. Therefore, this creates a computational bottleneck for time-sensitive application domains and devices with limited processing power. For example, in high-dimensional classification with hundreds or thousands of dimensions, computing the pairwise Euclidean distances between data points becomes a time-consuming task.

Memory requirements In modern data analysis, it is often prohibitive to store the entire data set in a central processing unit, which is required by many existing learning methods. In fact, the required amount of storage can easily exceed the available memory (RAM) of a typical desktop or mobile device. This problem is exacerbated in streaming data settings where data points are presented sequentially and are never stored. In this case, it is only possible to have a few passes (ideally one pass) over the full data set [100]. Therefore, memory limitations of processing units create a considerable challenge to the design and statistical analysis of data analytic tasks in today's high-dimensional data regimes.

Communication cost In distributed learning settings, data vectors are observed and stored in a number of distributed sites. For example, in applications such as environmental monitoring, sensors measure quantities of interest over geographically-distributed sites [15]. In this case, we would like to extract the underlying structure in a centralized fusion center which aggregates the data from all distributed sites [116]. It is often infeasible to communicate a massive high-dimensional data set among distributed sites due to communication constraints. Hence, efficient compression schemes will be indispensable to reduce the size of data and communication costs. However, the main challenge here is to extract information from the compressed data in a centralized fusion center [42].

The primary focus of this dissertation is centered around tackling these challenges and developing efficient signal processing and machine learning algorithms for large high-dimensional data. In particular, four scalable learning methods are introduced and studied,

which cover important linear and nonlinear techniques in data analysis. These problems are explained in Section 1.1. A detailed overview of randomized algorithms and their applications in large-scale data analysis are provided in Section 1.2. See Section 1.3 for our departure from existing randomized methods. Finally, the organization of this dissertation and summary of contributions are presented in Section 1.4.

1.1 Problems of Interest

In this section, we present a brief summary of signal processing and machine learning algorithms that are the main focus of this dissertation.

1.1.1 Principal component analysis

Principal component analysis (PCA) is a widely used statistical tool in data analysis and machine learning. It is frequently used for dimensionality reduction, feature extraction, and as a preprocessing step for classification in many applications such as face recognition [136] and hyperspectral imaging [142].

PCA assumes that a collection of n centered data vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ lie within or close to a r-dimensional subspace of $\mathbb{R}^p$. Consider the data matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ whose i-th column is the data sample $\mathbf{x}_i$. There are various ways to compute principal components of the data. One technique is based on forming the sample covariance matrix

$$\mathbf{C}_n = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T = \frac{1}{n} \mathbf{X}\mathbf{X}^T, \tag{1.1}$$

and computing the eigenvalue decomposition of the covariance matrix $\mathbf{C}_n$

$$\mathbf{C}_n = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T. \tag{1.2}$$

Here $\mathbf{\Lambda} = \mathrm{diag}([\lambda_1, \ldots, \lambda_p])$ is diagonal with $\lambda_1 \geq \ldots \geq \lambda_p \geq 0$ and $\mathbf{U} \in \mathbb{R}^{p \times p}$ contains the orthonormal eigenvectors. The first r columns of $\mathbf{U}$ are the principal components, which identify a r-dimensional subspace of $\mathbb{R}^p$ that best preserves the variance in the data.

Another approach to compute principal components is via the singular value decomposition (SVD) of the data matrix $\mathbf{X}$

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{1.3}$$

where the top left singular vectors are the desired principal components since we have

$$\mathbf{C}_n = \frac{1}{n}\mathbf{X}\mathbf{X}^T = \frac{1}{n}\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T = \mathbf{U}\left(\frac{1}{n}\mathbf{\Sigma}^2\right)\mathbf{U}^T. \tag{1.4}$$

In Fig. 1.1, a set of $n = 1000$ data vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{1000}$ in $\mathbb{R}^2$ are plotted. The top left singular vector of the data matrix $\mathbf{X}$ is also plotted, which represents the direction of maximum variance in the data.

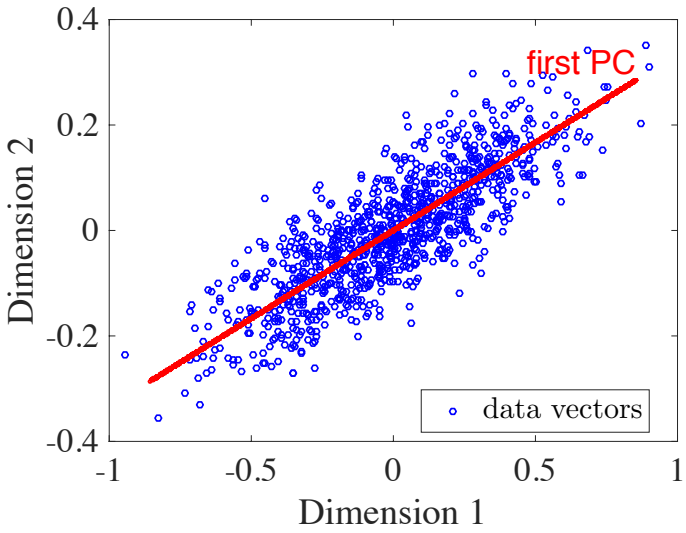


Figure 1.1: Visualization of the top left singular vector of the data matrix, i.e., first principal component (PC).

1.1.2 Sparse signal models and dictionary learning

Sparse signal models can be viewed as generalization of the linear subspace model and are able to capture more complex structure than principal component analysis [20]. In this framework, each data vector in the data set is allowed to use a different subset of basic

elements in order to achieve the best approximation. The sparse representation model has received a lot of attention in image processing, machine learning, and computer vision; see [97] for a comprehensive review.

Consider a data vector $\mathbf{x} \in \mathbb{R}^p$ and a matrix $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K] \in \mathbb{R}^{p \times K}$, known as a dictionary, that consists of K basic elements or atoms in $\mathbb{R}^p$. In the sparse representation model, our goal is to find a linear combination of a few elements from the dictionary $\mathbf{D}$ that is close to the data vector $\mathbf{x}$. This can be formulated as the following optimization problem

$$\widehat{\mathbf{c}} = \underset{\mathbf{c} \in \mathbb{R}^K}{\arg\min} \|\mathbf{x} - \mathbf{D}\mathbf{c}\|_2^2 \;\; s.t. \;\; \|\mathbf{c}\|_0 \leq T \tag{1.5}$$

where $\|\mathbf{c}\|_0$ counts the number of nonzero entries of the coefficient vector $\mathbf{c}$. This problem is often referred to as sparse coding and it is NP-hard in general. There are two groups of algorithms for finding an approximation solution to this problem. The first line of work known as basis pursuit replaces the ℓ_0 norm with ℓ_1, which promotes the sparsity of solution [30]. The second line of work finds an approximation solution by choosing the atoms one at a time in order to minimize the objective function in (1.5). A well-known example of this group is the Orthogonal Matching Pursuit (OMP) algorithm [134].

A key component of the sparse representation model is that the dictionary $\mathbf{D}$ should be inferred from the set of training data vectors. Given a set of n data vectors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ in $\mathbb{R}^p$, a dictionary $\mathbf{D} \in \mathbb{R}^{p \times K}$ that leads to the best representation under a strict sparsity constraint for each member in the set is obtained by minimizing the following representation error

$$\min_{\mathbf{D},\mathbf{C}} \|\mathbf{X} - \mathbf{D}\mathbf{C}\|_F^2 \;\; s.t. \;\; \forall i, \quad \|\mathbf{c}_i\|_0 \leq T \tag{1.6}$$

where $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_n]$ is the coefficient matrix corresponding to the training data $\mathbf{X}$. Solving this optimization problem may lead to an exact representation for the i-th data sample $\mathbf{x}_i$ such that $\mathbf{x}_i = \mathbf{D}\mathbf{c}_i$ or each $\mathbf{x}_i$ may be well-approximated by the learned dictionary, satisfying $\|\mathbf{x}_i - \mathbf{D}\mathbf{c}_i\|_2 \leq \epsilon$ for some small ϵ.

In [6], an iterative algorithm, K-SVD, was presented that alternates between sparse coding of the signals with respect to a fixed dictionary and the process of updating dictionary atoms. The main contribution of the K-SVD algorithm is the dictionary update process which minimizes the objective in (1.6) using a simple iterative approach. This objective can be written as

$$\begin{aligned} \|\mathbf{X} - \mathbf{DC}\|_F^2 &= \left\| \left(\mathbf{X} - \sum_{j \neq k} \mathbf{d}_j \mathbf{c}^j \right) - \mathbf{d}_k \mathbf{c}^k \right\|_F^2 \\ &= \left\| \mathbf{E}_k - \mathbf{d}_k \mathbf{c}^k \right\|_F^2, \end{aligned} \tag{1.7}$$

where $\mathbf{d}_j$ is the j-th dictionary atom, $\mathbf{c}^j \in \mathbb{R}^{1 \times n}$ is the corresponding coefficients (the j-th *row* of $\mathbf{C}$), and $\mathbf{E}_k := \mathbf{X} - \sum_{j \neq k} \mathbf{d}_j \mathbf{c}^j$ is the representation error for the training data when the k-th dictionary atom is removed.

In the dictionary update process, it is assumed at each step k that $\mathbf{d}_j$ and $\mathbf{c}^j$, $j \neq k$, are fixed. We then minimize the criterion over $\mathbf{d}_k$ and $\mathbf{c}^k$ which is equivalent to finding the best rank-1 approximation of $\mathbf{E}_k$. The minimizer might typically be obtained by applying the SVD to the matrix $\mathbf{E}_k$, but the strict sparsity constraint also must be considered. Therefore, in the K-SVD algorithm, we shrink the matrix $\mathbf{E}_k$ by eliminating columns corresponding to those training data points for which the elements in the k-th row of $\mathbf{C}$ are zero. The best rank-1 approximation of the shrunken $\mathbf{E}_k$ is then computed to update $\mathbf{d}_k$ and $\mathbf{c}^k$. This strategy preserves the support of the coefficient matrix $\mathbf{C}$.

In Fig. 1.2, 256 learned dictionary atoms via K-SVD [1] are shown for 8×8 natural image patches ($p = 64$).

K-SVD is one of the most popular algorithms for sparse representation and dictionary learning. Besides K-SVD, there are several other methods to learn discriminative dictionaries for the purpose of classification, such as Fisher Discriminant Dictionary Learning [146] and Label Consistent K-SVD [75].

[1] `http://www.cs.technion.ac.il/~elad/software/`

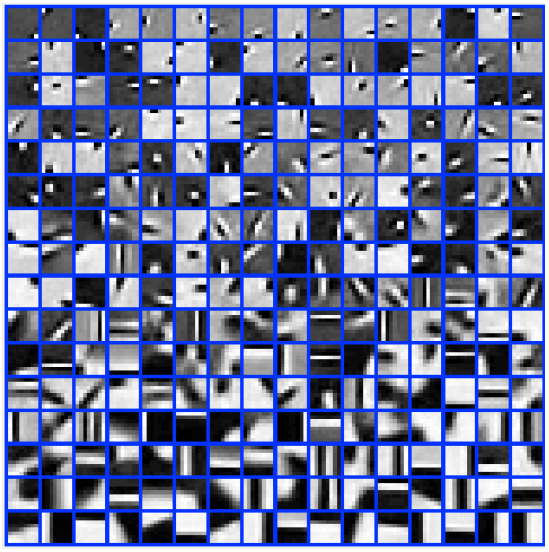


Figure 1.2: 256 learned dictionary atoms via K-SVD.

1.1.3 Clustering

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 bioinformatics [22, 101].

Among clustering algorithms, K-means is one of the most popular clustering algorithms [24]. More formally, K-means defines a clustering criterion, and Lloyd's algorithm [91] is a heuristic to solve it. Throughout the dissertation, we use "K-means" and "Lloyd's algorithm" synonymously. Consider a collection of n data samples in $\mathbb{R}^p$, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ represents the data matrix. The K-means algorithm splits the data set into a known number of K clusters. The resulting K-partition $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$ is a collection of K non-empty pairwise disjoint sets that covers the data set. Moreover, each cluster $\mathcal{S}_k$ is represented using a vector $\boldsymbol{\mu}_k \in \mathbb{R}^p$ that is associated with the k-th cluster.

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

$$J(\mathbf{c}, \boldsymbol{\mu}) = \sum_{i=1}^{n} \sum_{k=1}^{K} c_{ik} \left\| \mathbf{x}_i - \boldsymbol{\mu}_k \right\|_2^2 . \tag{1.8}$$

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:

$$\forall \;\; i = 1, \ldots, n : \;\; c_{ik} = \begin{cases} 1, & k = \arg\min_j \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|_2^2 \\ 0, & \text{otherwise} \end{cases} \tag{1.9}$$

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

$$\forall \; k = 1, \ldots K : \;\; \boldsymbol{\mu}_k = \frac{1}{n_k} \sum_{\mathbf{x}_i \in \mathcal{S}_k} \mathbf{x}_i \tag{1.10}$$

where $\mathcal{S}_k$ denotes the set of samples assigned to the k-th cluster in Step 1 and $n_k = |\mathcal{S}_k|$. Therefore, the update formula for cluster center $\boldsymbol{\mu}_k$ is the sample mean of the data samples in $\mathcal{S}_k$.

To gain some intuition, we consider a synthetic data set shown in Fig. 1.3. This data set consists of $n = 2000$ samples in $\mathbb{R}^2$. The standard K-means algorithm with a random initialization is employed to find the assignments as well as cluster centers with the parameter $K = 2$. The K-means algorithm finds a meaningful partition of the data set into two clusters $\mathcal{S}_1$ and $\mathcal{S}_2$. Like dictionary learning, solving K-means exactly is infeasible, but the K-means heuristic often works well.

1.1.4 Kernel machines

Kernel-based approaches are popular methods for supervised and unsupervised learning. Well-known examples include support vector machines (SVM) [38], kernel principal component analysis (KPCA) [121], kernel ridge regression [122], and kernel clustering [61].

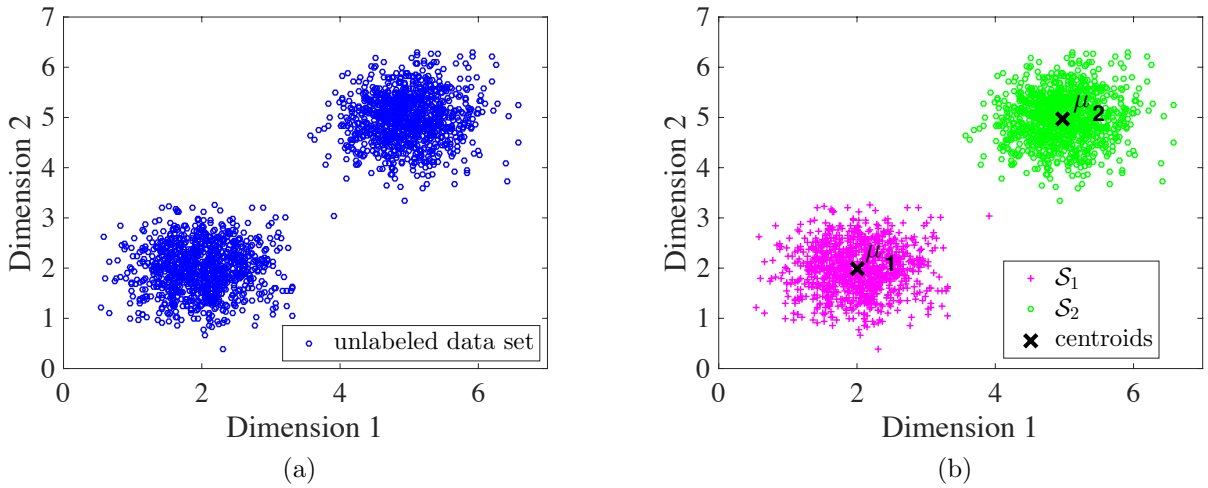


Figure 1.3: (a) Synthetic data set in $\mathbb{R}^2$ and (b) the resulting cluster centroids and assignments using the K-means algorithm.

The main idea behind kernel-based learning is to map the input data points into a feature space, where all pairwise inner products of the mapped data points can be computed via a nonlinear kernel function. Thus, kernel methods allow one to use linear algorithms in the higher (or infinite) dimensional feature space which correspond to nonlinear algorithms in the original space.

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ be a data matrix that contains n data points. The inner products in feature space are calculated using a "kernel function" $\kappa(\cdot, \cdot)$ defined on the original space

$$K_{ij} := \kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \quad \forall i, j = 1, \ldots, n, \tag{1.11}$$

where $\Phi : \mathbf{x} \mapsto \Phi(\mathbf{x})$ is the kernel-induced feature map. All pairwise inner products of the n mapped data points are stored in the so-called "kernel matrix" $\mathbf{K} \in \mathbb{R}^{n \times n}$, where the (i, j)-th entry is K_{ij}. Two well-known examples of kernel functions that lead to symmetric positive semidefinite kernel matrices are Gaussian and polynomial kernel functions. The former takes the form $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / c)$ and the polynomial kernel is of the form $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$, where $c \in \mathbb{R}^+$ and $d \in \mathbb{N}$ are the parameters [137, 111].

Kernel-based K-means clustering has gained popularity due to its simplicity and the power of its implicit nonlinear representation of the data. The Kernel K-means algorithm finds a K-partition of the mapped data $\{\Phi(\mathbf{x}_i)\}_{i=1}^n$ by minimizing

$$\mathcal{L}(\mathcal{S}) = \sum_{i=1}^{n}\sum_{k=1}^{K} c_{ik}\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_k\|_2^2. \tag{1.12}$$

The optimization problem of minimizing (1.12) can be solved using the same iterative procedure of K-means. To see this, consider the centroid of the j-th cluster

$$\boldsymbol{\mu}_j = \frac{1}{|\mathcal{S}_j|}\sum_{\Phi(\mathbf{x}_l)\in\mathcal{S}_j}\Phi(\mathbf{x}_l). \tag{1.13}$$

This centroid cannot be computed explicitly, but we can compute the distance between each mapped data sample $\Phi(\mathbf{x}_i)$ and the centroid using (1.13)

$$\begin{aligned}\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j\|_2^2 &= \langle\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j, \Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j\rangle \\ &= K_{ii} - \frac{2}{|\mathcal{S}_j|}\sum_{\Phi(\mathbf{x}_l)\in\mathcal{S}_j} K_{il} + \frac{1}{|\mathcal{S}_j|^2}\sum_{\Phi(\mathbf{x}_l),\Phi(\mathbf{x}_l')\in\mathcal{S}_j} K_{ll'}.\end{aligned} \tag{1.14}$$

Hence, we see that Kernel K-means is an iterative algorithm that requires access to the full kernel matrix $\mathbf{K}$ without the need to explicitly map the data points.

To demonstrate the performance of kernel-based clustering, consider a synthetic data set containing $n = 2000$ points in $\mathbb{R}^2$ shown in Fig. 1.4a. This data set is nonlinearly separable but not linearly separable. Thus, standard K-means is not able to identify these two clusters, and the two centroids selected by standard K-means do not describe the true clusters (Fig. 1.4b). To address this problem, we compute the kernel matrix $\mathbf{K}$ whose elements are obtained by using the polynomial kernel of order $d = 2$. Then, the singular value decomposition (SVD) of the kernel matrix $\mathbf{K}$ is computed to find the best rank-2 approximation of the kernel matrix in the form of $\mathbf{K} \approx \mathbf{L}\mathbf{L}^T$, where $\mathbf{L} \in \mathbb{R}^{2000\times 2}$ [108]. The rows of $\mathbf{L}$ are plotted in Fig. 1.4c; the transformed data points are linearly separable and the standard K-means algorithm provides a meaningful partition of these data points. The resulting assignments in the original input space are shown in Fig. 1.4d.

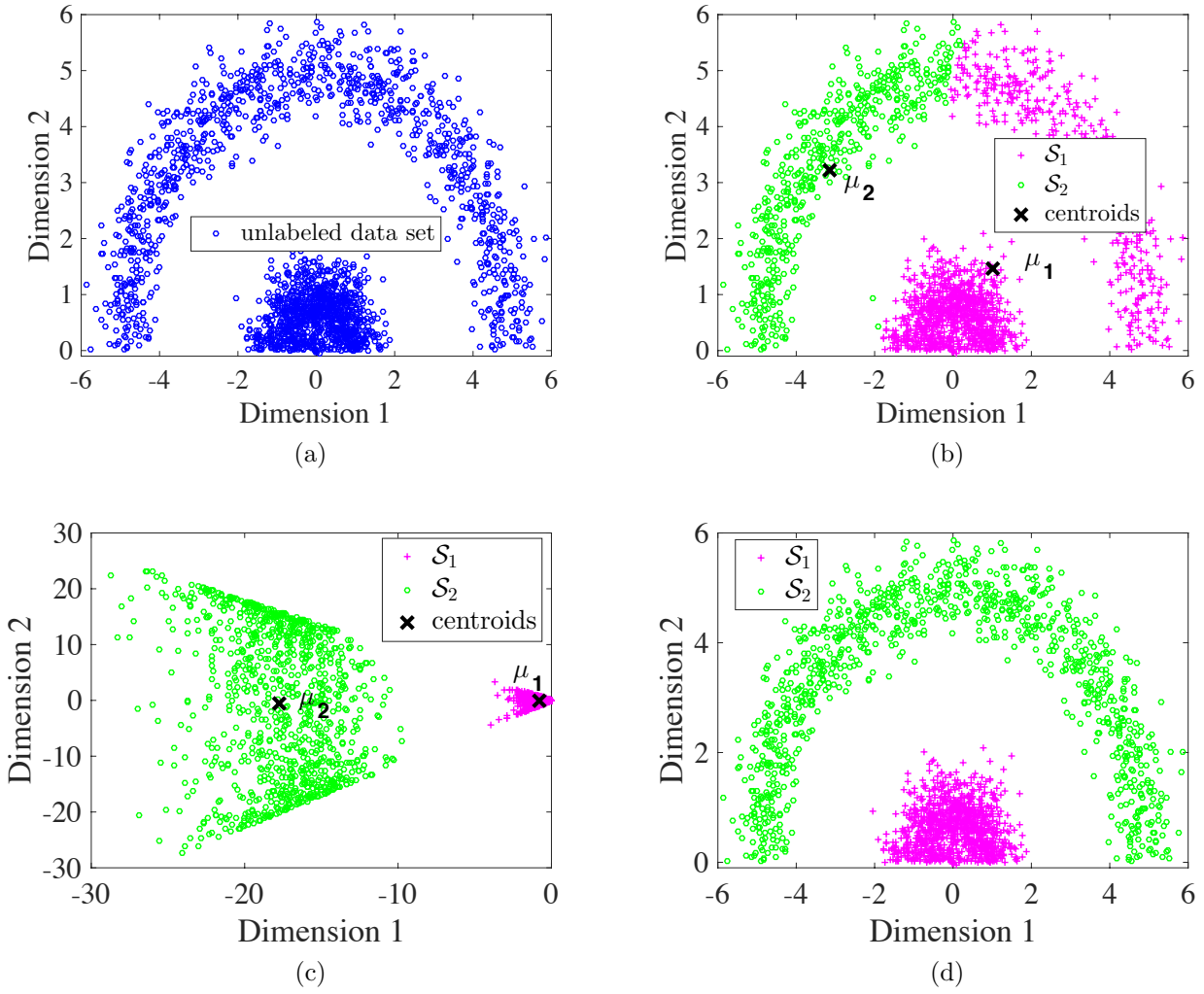


Figure 1.4: (a) Synthetic data set in $\mathbb{R}^2$, (b) the resulting cluster centroids and assignments using the K-means algorithm, (c) the resulting assignments and centroids in the transformed kernel space, and (d) the resulting assignments using Kernel K-means in the original input space.

1.2 Randomized Algorithms

Randomized dimension reduction methods have been proven to be powerful tools to tackle the challenges of modern large-scale data analysis. These methods use data-independent mappings which involve some randomness in order to embed high-dimensional data vectors into a lower dimensional space. This line of work is motivated by the ob-

servation that the intrinsic dimension of many data sets is significantly smaller than their ambient dimension. Thus, randomized methods seek to reduce the data dimension while preserving the underlying structures and patterns in high-dimensional data sets. The task of learning is then performed on low-dimensional summaries of the data instead of the original high-dimensional data [42, 117].

Considering linear data-independent maps (i.e., excluding PCA), we see randomness is essential. If we use a fixed deterministic map, then we lose information for all the vectors in its null space. We need randomness on either the map or the inputs in order to prevent catastrophic outcomes. The randomized scheme is effective in reducing the memory, computation, and communications costs of data analytic tasks in today's high-dimensional data regimes. In fact, compared with standard deterministic dimension reduction algorithms [39], randomized methods often obviate the need to store and manipulate an entire data set. Thus, randomized methods are valuable tools for analyzing modern data sets.

The main focus of this dissertation is centered around randomized dimensionality reduction techniques using random projections and random sampling. We explain these methods in Section 1.2.1. Furthermore, two important applications of randomized methods in signal processing and machine learning are discussed in Section 1.2.2. See [68, 95, 144] and the references therein for a comprehensive survey on the applications of randomized methods.

1.2.1 Random projections and random sampling

Random projections and random sampling are simple and effective dimensionality reduction techniques. Random projections are used to construct compressed versions of high-dimensional data by computing linear combinations of dimensions weighted by random coefficients. To be formal, consider a data vector $\mathbf{x} \in \mathbb{R}^p$ and a random projection matrix $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_m] \in \mathbb{R}^{p \times m}$, $m < p$, to project the data vector to a lower dimensional space

$$\mathbf{y} = \mathbf{R}^T \mathbf{x} \in \mathbb{R}^m. \tag{1.15}$$

This mapping is equivalent to taking the inner product of $\mathbf{x} \in \mathbb{R}^p$ with m random vectors $\mathbf{r}_1, \ldots, \mathbf{r}_m \in \mathbb{R}^p$, depicted in Fig. 1.5.

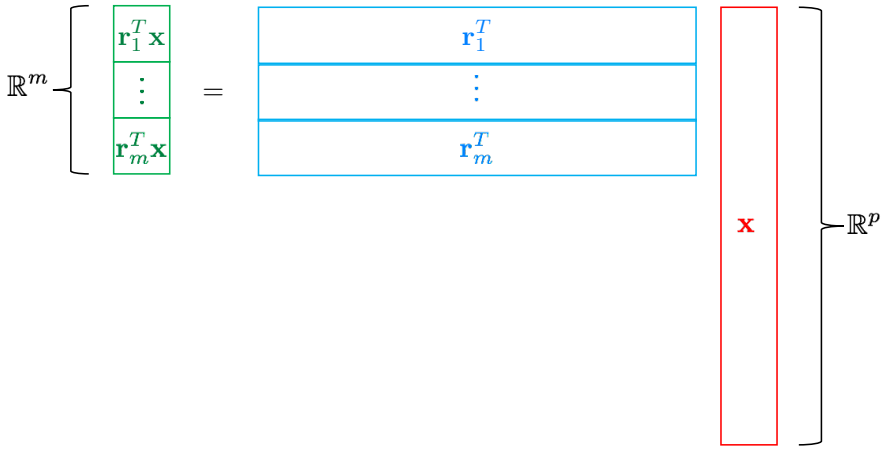


Figure 1.5: Random projection matrix $\mathbf{R}$ consisting of m random vectors $\mathbf{r}_1, \ldots, \mathbf{r}_m$ in $\mathbb{R}^p$ is used to construct m-dimensional sketch of high-dimensional vector $\mathbf{x} \in \mathbb{R}^p$.

There are different classes of random projection matrices for constructing low-dimensional sketches and we describe a few of them in this section. The most well-known sketch is based on a matrix $\mathbf{R} \in \mathbb{R}^{p \times m}$ with independent standard Gaussian entries, i.e.,

$$R_{ij} \sim \mathcal{N}(0,1), \quad \mathbb{E}[R_{ij}] = 0,\ \mathbb{E}[R_{ij}^2] = 1,\ \mathbb{E}[R_{ij}^4] = 3. \tag{1.16}$$

Gaussian random matrices have been widely used due to the nice properties of the Gaussian distribution, e.g., the standard Gaussian distribution is 2-stable, which simplify the theoretical analysis to show that the resulting sketches preserve certain structures. However, in practice, such matrices should be formed and manipulated explicitly. The high costs of storing and multiplying dense Gaussian matrices make them impractical for large high-dimensional data sets.

To alleviate this bottleneck, a general class of random matrices with i.i.d. entries drawn

from the following distribution was introduced in [87] for the sparsity parameter $s \geq 1$

$$R_{ij} = \begin{cases} +1 & \text{with prob. } \frac{1}{2s} \\ 0 & \text{with prob. } 1 - \frac{1}{s} \\ -1 & \text{with prob. } \frac{1}{2s} \end{cases}. \tag{1.17}$$

Therefore, the parameter s controls the sparsity of random matrices such that each column of $\mathbf{R}$ has $\frac{p}{s}$ nonzero entries, on average. If we set the parameter $s = 1$, the resulting random matrix whose entries have values ± 1 uniformly at random is called a random sign matrix. The first four moments of this distribution can be easily computed as follows

$$\mathbb{E}[R_{ij}] = \mathbb{E}[R_{ij}^3] = 0, \;\; \mathbb{E}[R_{ij}^2] = \mathbb{E}[R_{ij}^4] = \frac{1}{s}. \tag{1.18}$$

The sparse random matrices can be parameterized by an important statistical quantity which measures the heaviness of tail for a distribution. The kurtosis is defined as

$$\kappa := \frac{\mathbb{E}[R_{ij}^4]}{\mathbb{E}[R_{ij}^2]^2} - 3, \tag{1.19}$$

and, by definition, the kurtosis of the standard Gaussian distribution is zero. For the sparse Rademacher matrix defined in (1.17), we get $\kappa = s - 3$ which can be viewed as a measure of deviation from the Gaussian distribution as the sparsity parameter s increases.

Another approach for constructing sketches of high-dimensional data is based on random sampling or selection. Let $\mathbf{e}_j$ denote the j-th vector of the canonical basis in $\mathbb{R}^p$, where entries are all zero except for the j-th one which is 1. In random sampling, each column of the random matrix $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_m] \in \mathbb{R}^{p \times m}$ is chosen from the set of p canonical basis vectors using a given probability distribution $\{\zeta_j\}_{j=1}^p$. Thus, the i-th column of the random matrix $\mathbf{R}$ takes on the values

$$\mathbf{r}_i = \mathbf{e}_j \qquad \text{with probability } \; \zeta_j \; \text{ for } \; j = 1, \ldots, p. \tag{1.20}$$

Note that each column of such random matrices has exactly one nonzero entry and, thus, the matrix $\mathbf{R}$ is never explicitly formed nor stored. For example, when $\zeta_1 = \ldots = \zeta_p = \frac{1}{p}$, the mapping $\mathbf{R}^T\mathbf{x} \in \mathbb{R}^m$ keeps m entries of the vector $\mathbf{x} \in \mathbb{R}^p$ uniformly at random.

It is shown that nonuniform distributions $\{\zeta_j\}_{j=1}^{p}$ based on leverage scores [47] may improve the performance of learning methods using low-dimensional sketches. However, such techniques are more complicated than the uniform sampling because the left singular vectors of the data matrix should be estimated in order to estimate the leverage score.

1.2.2 Applications

In this section, we review two important applications of randomized dimension reduction in large-scale data analysis. In particular, we summarize existing results on the choice of m in order to ensure that randomized mappings preserve quantities of interest. Theoretical results that guarantee the effectiveness of randomized methods come with probabilistic bounds which assert the outcome of a specific randomized method is close to the desired outcome with high probability.

1.2.2.1 Preserving pairwise Euclidean distances

One of the key applications of randomized algorithms is based on the celebrated Johnson-Lindenstrauss (JL) lemma [76]; any n data vectors can be embedded into $\mathbb{R}^m$ with $m = \mathcal{O}(\varepsilon^{-2} \log n)$ so that all the pairwise Euclidean distances are preserved within a multiplicative factor of $(1 \pm \varepsilon)$. The JL lemma is an effective method for speeding up solutions to many high-dimensional problems and reducing the amount of storage in streaming data settings. The JL lemma has been used in many applications such as classification of high-dimensional data [7], numerical linear algebra [68], and compressive sensing [21, 82].

To explain the JL lemma, we start with the definition of the JL transform. In the following, a random matrix $\mathbf{R}$ is applied to a vector $\mathbf{x} \in \mathbb{R}^p$, which can be viewed as $\mathbf{x} = \mathbf{x}_i - \mathbf{x}_j$ for two data vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathbb{R}^p$.

Definition 1.1. *A random matrix $\mathbf{R} \in \mathbb{R}^{p \times m}$ forms a Johnson-Lindenstrauss transform if,*

for any fixed vector $\mathbf{x} \in \mathbb{R}^p$ *and any* $0 < \varepsilon < 1$*, we have*

$$\mathbb{P}\left((1-\varepsilon)\|\mathbf{x}\|_2^2 \leq \|\mathbf{R}^T\mathbf{x}\|_2^2 \leq (1+\varepsilon)\|\mathbf{x}\|_2^2\right) \geq 1 - \exp\left(-Cm\varepsilon^2\right), \tag{1.21}$$

where $C > 0$ *is an absolute constant.*

Based on the above definition, the parameter m should be chosen

$$m = \mathcal{O}\left(\frac{\log(1/\delta)}{\varepsilon^2}\right), \tag{1.22}$$

to achieve failure probability at most δ. Note that the mapping is into a dimension that is independent of the original dimension p.

There are different classes of random projection matrices that possess the JL transform property. For example, a rescaled random sign matrix with entries drawn uniformly at random from $\{\pm 1/\sqrt{m}\}$ preserves the pairwise Euclidean distances, as stated in the following theorem from [1, Theorem 1.1] and [26, Theorem 4].

Theorem 1.1. *Let* $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p\times n}$ *represent a set of* n *data points in* $\mathbb{R}^p$*. Let* $\mathbf{R} \in \mathbb{R}^{p\times m}$ *be a rescaled sign matrix with*

$$m = \frac{36}{\varepsilon^2}\log(n)\log(1/\delta). \tag{1.23}$$

Then, for all $i, j \in \{1, \ldots, n\}$*, we have the following property*

$$(1-\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|\mathbf{R}^T(\mathbf{x}_i - \mathbf{x}_j)\|_2^2 \leq (1+\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \tag{1.24}$$

with probability at least $1 - \delta$ *with respect to the randomness of the matrix* $\mathbf{R}$*.*

1.2.2.2 Randomized K-means clustering

Consider a set of n data points $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p\times n}$. The goal of K-means algorithm is to compute the optimal partitioning of the data set into K clusters

$$\mathcal{S}_{opt} = \arg\min_{\mathcal{S}} \mathcal{L}(\mathbf{X}, \mathcal{S}), \tag{1.25}$$

where $\mathcal{L}(\mathbf{X}, \mathcal{S}) := \sum_{i=1}^{n} \sum_{k=1}^{K} c_{ik} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$, cf. Section 1.1.3. To reduce the memory and computational complexity of K-means, one approach is to first reduce the dimensionality of data set via the random mapping $\mathbf{Y} = \mathbf{R}^T\mathbf{X} = [\mathbf{R}^T\mathbf{x}_1, \ldots, \mathbf{R}^T\mathbf{x}_n] \in \mathbb{R}^{m \times n}$, based on (1.15). Then, the K-means algorithm will be performed on the reduced data $\mathbf{y}_1, \ldots, \mathbf{y}_n$ in $\mathbb{R}^m$ to find the optimal K-partition

$$\widehat{\mathcal{S}}_{opt} = \arg\min_{\mathcal{S}} \mathcal{L}(\mathbf{Y}, \mathcal{S}). \tag{1.26}$$

Since the K-means clustering problem is NP-hard, we consider a class of approximate algorithms based on the following definition.

Definition 1.2. *Let $\mathcal{S}_{opt}$ and $\mathcal{L}(\mathbf{X}, \mathcal{S}_{opt})$ denote the optimal partitioning and the optimal objective value of K-means, respectively. An algorithm is called a "θ-approximation" for the K-means clustering algorithm with the parameter $\theta \geq 1$ if it returns a partitioning $\mathcal{S}_\theta$ such that with probability at least $1 - \delta_\theta$, the following holds*

$$\mathcal{L}(\mathbf{X}, \mathcal{S}_\theta) \leq \theta \cdot \mathcal{L}(\mathbf{X}, \mathcal{S}_{opt}). \tag{1.27}$$

Therefore, in practice, a θ-approximation K-means algorithm is performed on the sketches of high-dimensional data to obtain the K-partition

$$\widehat{\mathcal{S}}_\theta \approx \arg\min_{\mathcal{S}} \mathcal{L}(\mathbf{Y}, \mathcal{S}), \tag{1.28}$$

where $\widehat{\mathcal{S}}_\theta$ is the partition obtained after running the θ-approximation K-means clustering algorithm on $\mathbf{y}_1, \ldots, \mathbf{y}_n$. The overall procedure is depicted in Fig. 1.6.

A straightforward application of JL lemma from Section 1.2.2.1 can be used to study the approximation error of K-means clustering on the reduced data. Let $\mathbf{R} \in \mathbb{R}^{p \times m}$ be a random matrix that satisfies the JL transform; all n data points are projected into $\mathbb{R}^m$ with $m = \mathcal{O}(\log(n)/\varepsilon^2)$ so that the Euclidean pairwise distances are preserved within a multiplicative factor $(1 \pm \varepsilon)$. The K-means objective function depends only on pairwise

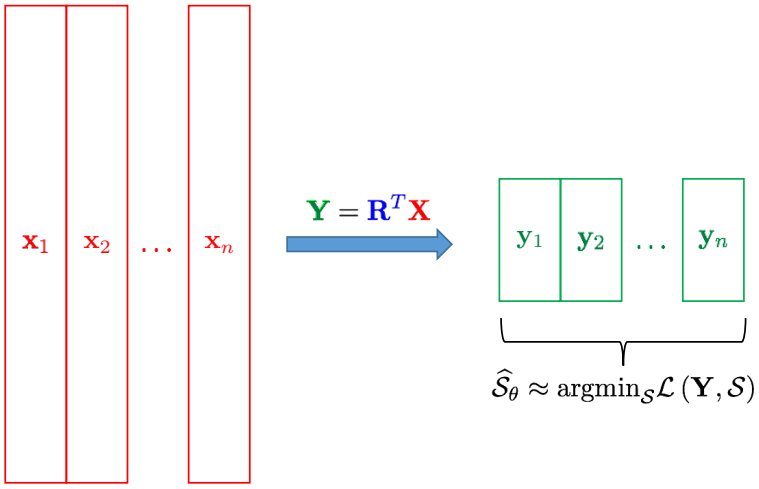


Figure 1.6: Running a θ-approximation K-means clustering algorithm on the reduced data based on random projections.

distances of the data vectors from the corresponding centroids, which means that with high probability

$$\mathcal{L}\left(\mathbf{X}, \widehat{\mathcal{S}}_{opt}\right) \leq (1+\varepsilon) \cdot \mathcal{L}\left(\mathbf{X}, \mathcal{S}_{opt}\right). \tag{1.29}$$

Therefore, finding the optimal K-partition in the low-dimensional space and plugging it back to partition the original high-dimensional data result in a $(1+\varepsilon)$-approximation clustering algorithm with the parameter $m = \mathcal{O}(\log(n)/\varepsilon^2)$.

In a recent work, Mahoney *et al.* [26] presented a new analysis of randomized K-means clustering. It was shown that a smaller number of dimensions m are enough to achieve an accurate clustering algorithm. In fact, the new analysis reveals that the task of partitioning n data vectors into K clusters requires $m = \mathcal{O}(K/\varepsilon^2)$ dimensions, which removes the dependence on the number of data vectors n. This improved result is significant for modern data sets with massive sample sizes. In the following, we restate [26, Theorem 2].

Theorem 1.2. *Consider a set of n data points in $\mathbb{R}^p$, i.e., $\mathbf{X} \in \mathbb{R}^{p \times n}$. Let K be the number of clusters for K-means clustering and $\mathbf{R} \in \mathbb{R}^{p \times m}$ be a rescaled sign matrix with $m = \mathcal{O}(K/\varepsilon^2)$.*

Run any θ-approximation K-means clustering algorithm with failure probability δ_θ on $\mathbf{Y} = \mathbf{R}^T\mathbf{X} \in \mathbb{R}^{m\times n}$ to obtain $\widehat{\mathcal{S}}_\theta$, c.f. Fig. 1.6. Then, for $\varepsilon \in (0, 1/3)$ with probability at least $0.96 - \delta_\theta$

$$\mathcal{L}\left(\mathbf{X}, \widehat{\mathcal{S}}_\theta\right) \leq (1 + (1+\varepsilon)\theta) \cdot \mathcal{L}\left(\mathbf{X}, \mathcal{S}_{opt}\right). \tag{1.30}$$

To gain some intuition, let us consider $\theta = 1$, i.e., $\widehat{\mathcal{S}}_\theta$ is the optimal K-partition. Finding the optimal partitioning of the reduced data in $\mathbb{R}^m$ with $m = \mathcal{O}(K/\varepsilon^2)$ and plugging back the resulting K-partition in the original high-dimensional space yield a $(2+\varepsilon)$-approximation randomized clustering algorithm.

1.3 A Randomized Scheme with Multiple Random Matrices

This dissertation departs from the prior work on randomized dimension reduction in that independent and distinct random matrices $\mathbf{R}_i \in \mathbb{R}^{p\times m}$ are considered for each data vector $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1, \ldots, n$. Most existing randomized algorithms use a *shared* random matrix across data vectors in a data set, cf. Fig. 1.6. However, the assumption of multiple and distinct $\mathbf{R}_i$ is crucial for consistent and accurate recovery of important quantities, such as principal components or dictionary atoms, from sketches of high-dimensional data. The dimension reduction technique with distinct random matrices is shown in Fig. 1.7.

In this section, we consider a simple intuitive example to clarify the importance of multiple random matrices. Let us consider the following probabilistic generative model for generating a set of n data points

$$\mathbf{x}_i = \overline{\mathbf{x}} + \mathbf{n}_i, \quad i = 1, \ldots, n, \tag{1.31}$$

where $\overline{\mathbf{x}} \in \mathbb{R}^p$ is a fixed vector and $\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$ is a multivariate normal variable. The sample mean estimator is defined as

$$\widehat{\overline{\mathbf{x}}}_n := \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i. \tag{1.32}$$

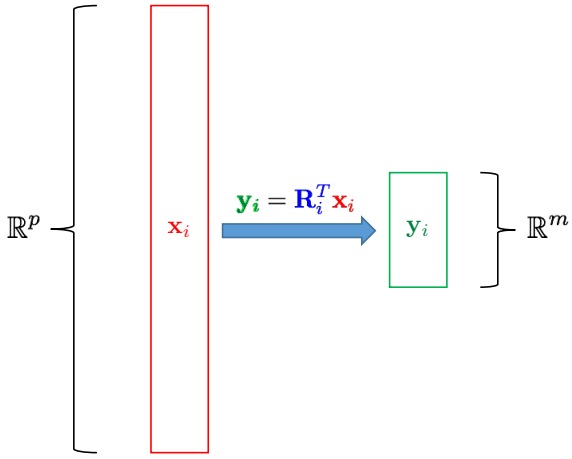


Figure 1.7: Consider a set of high-dimensional data vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^p$. Independent and distinct random matrices $\mathbf{R}_i \in \mathbb{R}^{p \times m}$ are used to construct sketches $\mathbf{y}_i = \mathbf{R}_i^T \mathbf{x}_i \in \mathbb{R}^m$.

This estimator converges to $\overline{\mathbf{x}}$ almost surely as $n \to \infty$.

Now, we would like to estimate $\overline{\mathbf{x}}$ from sketches of high-dimensional data in order to compare two cases: (1) *shared* random matrix $\mathbf{R}$ for all data vectors, and (2) *distinct* random matrices $\mathbf{R}_i$ for each $\mathbf{x}_i$, $i = 1, \ldots, n$. Assume that each column of the $p \times m$ random matrix $\mathbf{R}$ (or $\mathbf{R}_i$) is drawn uniformly at random from the set of canonical basis vectors in $\mathbb{R}^p$.

For the case with shared random matrix, the collected data are $\mathbf{y}_i = \mathbf{R}^T \mathbf{x}_i$, $i = 1, \ldots, n$, so the obvious estimator is

$$\widehat{\overline{\mathbf{x}}}_n^S := \frac{1}{n} \sum_{i=1}^{n} \mathbf{R}\mathbf{y}_i = \frac{1}{n} \mathbf{R}\mathbf{R}^T \sum_{i=1}^{n} \mathbf{x}_i \tag{1.33}$$

since $\left(\mathbf{R}^T\right)^\dagger = \mathbf{R}$, where $\dagger$ represents the pseudo-inverse. As $n \to \infty$, this estimator does not converge to $\overline{\mathbf{x}}$ because $\mathbf{R}\mathbf{R}^T \in \mathbb{R}^{p \times p}$ has rank at most $m < p$. Therefore, even with more data, we cannot achieve more accurate estimates because a random low-rank matrix is used to compress all the data. Hence, $\widehat{\overline{\mathbf{x}}}_n^S$ is not consistent.

However, we will present an unbiased sample mean estimator in Section 4 for the case that distinct random matrices $\mathbf{R}_i$, $i = 1, \ldots, n$, are employed:

$$\widehat{\overline{\mathbf{x}}}_n^D := \frac{p}{m} \frac{1}{n} \sum_{i=1}^{n} \mathbf{R}_i \mathbf{y}_i = \frac{p}{m} \frac{1}{n} \sum_{i=1}^{n} \mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i. \tag{1.34}$$

It is shown that this estimator converges to $\overline{\mathbf{x}}$ as $n \to \infty$, cf. Theorem 4.2. This is mainly due to the result that shows a rescaled version of $\sum_{i=1}^{n} \mathbf{R}_i \mathbf{R}_i^T \in \mathbb{R}^{p \times p}$ converges to the identity matrix $\mathbf{I}_p$, although each term $\mathbf{R}_i \mathbf{R}_i^T$ has rank at most m. Therefore, randomized dimensionality reduction with multiple random matrices leads to consistent recovery of the sample mean.

Moreover, we present a numerical simulation with parameters $p = 50$ and $m = 0.2 \times p = 10$. The normalized estimation error for various values of n averaged over 100 trials is reported in Fig. 1.8. As expected, the sample mean estimator based on a shared random matrix does not converge to the true mean as n increases. However, our approach is consistent since it uses a distinct random matrix for each $\mathbf{x}_i$. Thus, the set of all random low-rank matrices capture information about the entire space $\mathbb{R}^p$.

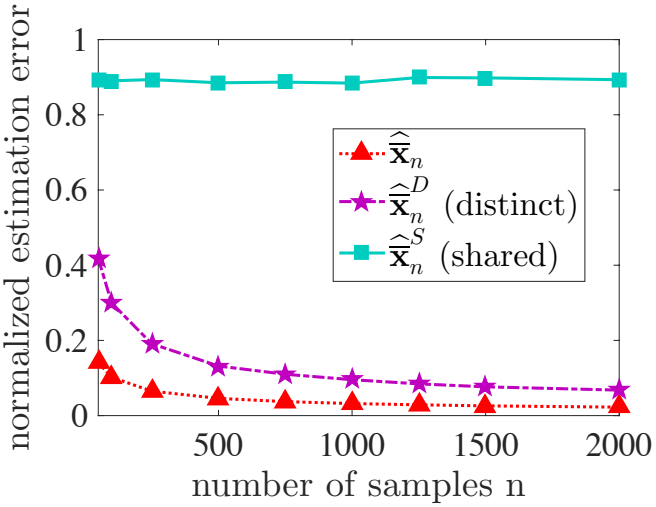


Figure 1.8: Normalized estimation error vs. number of samples n. As n increases, our proposed sample mean estimator based on distinct random matrices returns more accurate estimates.

As a final note, the same argument holds for random matrices with i.i.d. entries drawn from a centered distribution. Related results are presented in chapters 2 and 3.

1.4 Structure of Dissertation and Overview of Contributions

This dissertation consists of four main chapters, three of which have previously appeared as published articles in refereed conferences and journals. In this section, we will briefly summarize the main contributions in each chapter.

Chapter 2: The material in this chapter has appeared as:

- Pourkamali-Anaraki, F. (2016). Estimation of the sample covariance matrix from compressive measurements. IET Signal Processing, 10(9), 1089–1095.

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

- Pourkamali-Anaraki, F., and Hughes, S. (2014). Efficient recovery of principal components from compressive measurements with application to Gaussian mixture model estimation. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2332–2336.

Most of Chapter 2 considers a probabilistic generative model based on the ICML paper [113]. Section 2.7 provides a distribution-free analysis, which is published in IET Signal Processing [107].

In this chapter, we propose an approach to principal component estimation that utilizes projections onto very sparse random vectors with Bernoulli-generated nonzero entries (1.17). Indeed, our approach is simultaneously efficient in memory/storage space, efficient in computation, and produces accurate PC estimates, while also allowing for rigorous theoretical performance analysis. Moreover, one can tune the sparsity of the random vectors deliberately to achieve a desired point on the tradeoffs between memory, computation, and accuracy. We rigorously characterize these tradeoffs and provide statistical performance guarantees. In

addition to these very sparse random vectors, our analysis also applies to more general random projections. We present experimental results demonstrating that this approach allows for simultaneously achieving a substantial reduction of the computational complexity and memory/storage space, with little loss in accuracy, particularly for very high-dimensional data.

Chapter 3: The material in this chapter has appeared as:

- Pourkamali-Anaraki, F., Becker, S., and Hughes, S. (2015). Efficient dictionary learning via very sparse random projections. In Sampling Theory and Applications (SampTA), 478–482.
- Pourkamali-Anaraki, F., Hughes, S. (2013). Compressive K-SVD. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 5469–5473.

In this chapter, we extend previous work on compressive dictionary learning by showing that more general random projections may be used, including sparse ones. More precisely, we examine compressive K-means clustering as a special case of compressive dictionary learning and give theoretical guarantees for its performance for a very general class of random projections. We then propose a memory and computation efficient dictionary learning algorithm, specifically designed for analyzing large volumes of high-dimensional data, which learns the dictionary from very sparse random projections (1.17). Experimental results demonstrate that our approach allows for reduction of computational complexity and memory/data access, with controllable loss in accuracy.

Chapter 4: The material in this chapter has appeared as:

- Pourkamali-Anaraki, F., and Becker, S. (2017). Preconditioned Data Sparsification for Big Data with Applications to PCA and K-means. IEEE Transactions on Information Theory.

In this chapter, 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.

Chapter 5: The material in this chapter has not yet been published, but it is available online:

- Pourkamali-Anaraki, F., and Becker, S. (2016). Randomized Clustered Nyström for Large-Scale Kernel Machines. arXiv preprint: `https://arxiv.org/abs/1612.06470`.

The Nyström method has been popular for generating the low-rank approximation of kernel matrices that arise in many machine learning problems. The approximation quality of the Nyström method depends crucially on the number of selected landmark points and the selection procedure. In this chapter, we present a novel algorithm to compute the optimal Nyström low-approximation when the number of landmark points exceed the target rank. Moreover, we introduce a randomized algorithm for generating landmark points that is scalable to large-scale data sets. The proposed method performs K-means clustering on low-dimensional random projections of a data set and, thus, leads to significant savings for high-dimensional data sets. Our theoretical results characterize the tradeoffs between the accuracy and efficiency of our proposed method. Extensive experiments demonstrate the competitive performance as well as the efficiency of our proposed method.

Chapter 2

Memory and Computation Efficient PCA via Very Sparse Random Projections

2.1 Introduction

Principal component analysis (PCA) [78] is a fundamental tool in unsupervised learning and data analysis that finds the low-dimensional linear subspace that minimizes the mean-squared error between the original data and the data projected onto the subspace. The principal components (PCs) can be obtained by a singular value decomposition (SVD) of the data matrix or eigendecomposition of the data's covariance matrix. PCA is frequently used for dimensionality reduction, feature extraction, and as a pre-processing step for learning and recognition tasks such as classification.

There is a wealth of existing literature that develops computationally efficient approaches to computing these PCs. However, the overwhelming majority of this literature assumes ready access to the stored full data samples.

However, this full data access is not always possible in modern data settings. Modern data acquisition capabilities have increased massively in recent years, which can lead to a wealth of rapidly changing high-dimensional data. For example, in very large database environments, it may not be either feasible or practical to access all the data in storage. This "data deluge" creates an urgent need for scalable statistical learning tools that can gain insights from massive amounts of high-dimensional data [79, 127].

Moreover, in applications such as sensor networks, distributed databases, and surveillance, data is typically distributed over many sensors. Accessing all the data at once requires

tremendous communication costs between the sensors and a central processing unit. Algorithms that don't require access to all the data can help reduce this communication cost [115, 19]. A third case is streaming data, where one must acquire and store the data in real time to have full access, which may not be feasible.

One promising strategy to address these issues in a computationally efficient way, which also allows for rigorous theoretical analysis, is to use very sparse random projections. Random projections provide informative lower-dimensional representations of high-dimensional data, thereby saving memory and computation. They are widely used in many applications, including databases and data stream processing [87, 72], compressive sensing [45, 27], and low-rank matrix approximation [95].

Initial attempts have been made to perform PCA using only the information embedded in random projections. Unfortunately, however, theoretical guarantees have generally only been given for random vectors with i.i.d. entries drawn from the Gaussian distribution. This common choice is convenient in terms of theoretical analysis, but undesirable in practice. Such dense random vectors require relatively high storage space, and high computation because of the large amount of floating point arithmetic needed to compute each projection.

In this chapter, we instead aim to recover PCs from very sparse random projections with Bernoulli entries of ± 1. These sparse random projections can be implemented using simple database operations. Our theoretical analysis begins by assuming a probabilistic generative model for the data, related to the spiked covariance model. We then show that PCs computed from very sparse random projections are close estimators of the true underlying PCs. Moreover, one can adjust the sparsity of the random projections to reduce memory and computation as desired (at the cost of some accuracy). We give a rigorous analysis of the resulting tradeoffs between memory, computation, and accuracy as we vary sparsity, showing that efficiency in memory and computation may be gained with little sacrifice in accuracy. In fact, our analysis will also apply more generally to any random projections with i.i.d. zero mean and skewness entries with bounded second-, fourth-, sixth-, and eighth-order

moments, although we focus on the sparse-Bernoulli case.

In Section 2.2, we present a brief review of related work. The model assumptions and notation are in Section 2.3. We present an overview of the main contributions in Section 2.4. In Section 2.5, the main results are stated with some discussion of their consequences, with proofs deferred to the Appendices. Finally, we present experimental results demonstrating the performance and efficiency of our approach compared with prior work in Section 2.6. In Section 2.7, we consider a non-Bayesian data setting, which requires no distributional assumptions on the set of data samples.

2.2 Related Work

Algorithms that can efficiently recover PCs from a collection of full data samples have been an important topic in the literature for some time. This includes several lines of work. The first involves techniques that are based on dimensionality reduction, sketching, and subsampling for low-rank matrix approximation such as [95, 68, 67] and the references therein. In these methods, the computational complexity is typically reduced by performing the SVD on the smaller matrix obtained by sketching or subsampling. However, these methods require accessible storage of the entire data. This may not be practical for modern data processing applications where data samples are too vast or generated too quickly to be stored accessibly.

The second line of work involves online algorithms specifically tailored to have extremely low-memory complexity such as [141, 12] and the references therein. Typically, these algorithms assume that the data are streaming in continuously, that real-time PC estimates are needed, and they estimate the PCs by solving a stochastic optimization problem, in which each arriving data sample is used to update the PCs in an iterative procedure. As a couple recent examples of this line of work, in [100], it is shown that a blockwise stochastic variant of the power method can recover PCs in this low-memory setting from $\mathcal{O}(p \log p)$ samples, although the computational cost is not examined. Meanwhile, the authors in [13] develop an algorithm for learning PCs from streaming data, bound its generalization error

to new data, and also analyze its computational cost.

Our problem lies somewhere between the above two lines of work. We don't assume that memory/data access is not a concern, but at the same time, we also don't assume the extremely restrictive setting where one-sample-at-a-time real-time PC updates are required. Instead, we aim to reduce both memory and computation simultaneously for PCA across a broad class of big data settings, e.g. for enormous databases where loading into local memory may be difficult or costly, for streaming data when PC estimates do not have to be real-time, or for distributed data. We also aim to provide tunable tradeoffs for the amount of accuracy that will be sacrificed for each given reduction in memory/computation, in order to aid in choosing a desired balance point between these.

To do this, we recover PCs from random projections. There have been some related prior attempts to extract PCs from random projections of data [57, 114, 112]. In these papers, the problem of recovering PCs from random projections has been considered only for dense Gaussian random projections. However, dense vectors are undesirable for practical applications since they require relatively high storage space and computation (including lots of floating point arithmetic) as noted in the introduction. Our work will make use of sparse random vectors with Bernoulli entries which will be more efficiently implementable in a large database environment.

One paper that does use general sub-Gaussian random projections, but to sense an entire covariance matrix, rather than individual data samples, is [31]. However, guarantees are given only for the case of infinite data samples, making it hard to realistically use these results in memory/computation vs. accuracy tradeoffs, and computational cost is not examined. We will address both these issues.

As a final note, we observe that our work also can be viewed as an example of emerging ideas in computational statistics (see [28]) in which tradeoffs between computational complexity, dataset size, and estimation accuracy are explicitly characterized, so that a user may choose to reduce computation in very high-dimensional data settings with knowledge of the

risk to the accuracy of the result.

2.3 Problem Formulation and Notation

In this chapter, we focus on a statistical model for the data that is applicable to various scenarios. Assume that our original data in $\mathbb{R}^p$ are centered at $\overline{\mathbf{x}} \in \mathbb{R}^p$ and $\{\mathbf{v}_i\}_{i=1}^d \in \mathbb{R}^p$ are the d orthonormal principal components (PCs). We consider the following probabilistic generative model for the data samples

$$\mathbf{x}_i = \overline{\mathbf{x}} + \sum_{j=1}^{d} w_{ij}\sigma_j \mathbf{v}_j + \mathbf{z}_i,\ i = 1, \ldots, n \tag{2.1}$$

where the coefficients $\{w_{ij}\}$ are drawn i.i.d. from a zero mean distribution with variance 1, and $\{\mathbf{z}_i\}_{i=1}^n \in \mathbb{R}^p$ are drawn i.i.d. from $\mathcal{N}(\mathbf{0}, \frac{\epsilon^2}{p}\mathbf{I}_{p\times p})$ so that $\mathbb{E}[\|\mathbf{z}_i\|_2^2] = \epsilon^2$. Also, $\{\sigma_j\}_{j=1}^d$ are constants reflecting the energy of the data in each principal direction such that $\sigma_1 > \sigma_2 > \ldots > \sigma_d > 0$. The additive noise term $\mathbf{z}_i$ allows for some error in our assumptions. Note that the underlying covariance matrix of the data (excluding the noise term) is $\mathbf{C}_{true} \triangleq \sum_{j=1}^d \sigma_j^2 \mathbf{v}_j \mathbf{v}_j^T$, and the signal-to-noise ratio is SNR$= \frac{h}{\epsilon^2}$, where $h \triangleq \sum_{j=1}^d \sigma_j^2$. We will consider σ to be unknown, hence the term SNR. In fact, this model is related to the spiked covariance model [77] in which the data's covariance matrix is assumed to be a low-rank perturbation of the identity matrix.

We then introduce a very general class of random projections. Assume that matrices $\{\mathbf{R}_i\}_{i=1}^n \in \mathbb{R}^{p\times m}$, $m < p$, are formed by drawing each of their i.i.d. entries from a distribution with k^{th} order moments, μ_k, satisfying $\mu_1 = \mu_3 = 0$ and $\mu_k < \infty$ for $k = 2, 4, 6, 8$. In particular, we will be interested in a popular class of sparse random matrices, but our analysis will apply to any distribution satisfying these assumptions.

Each random projection $\mathbf{y}_i \in \mathbb{R}^m$ is then obtained by taking inner products of the data sample $\mathbf{x}_i \in \mathbb{R}^p$ with the random vectors comprising the columns of $\mathbf{R}_i$, i.e. $\mathbf{y}_i = \mathbf{R}_i^T \mathbf{x}_i$. Our goal is to provide theoretical guarantees for estimating the center and PCs of $\{\mathbf{x}_i\}_{i=1}^n$ from these random projections.

2.4 Overview of Contributions

In this work, we introduce two estimators for the center and underlying covariance matrix of data $\{\mathbf{x}_i\}_{i=1}^n$ from random projections $\{\mathbf{y}_i{=}\mathbf{R}_i^T\mathbf{x}_i\}_{i=1}^n$. In typical PCA, the center is estimated using the empirical center $\overline{\mathbf{x}}_{emp}{=}\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$. PCs are then obtained by eigendecomposition of the empirical covariance matrix $\mathbf{C}_{emp}{=}\frac{1}{n}\sum_{i=1}^n(\mathbf{x}_i-\overline{\mathbf{x}}_{emp})(\mathbf{x}_i-\overline{\mathbf{x}}_{emp})^T$. When $n \gtrsim p\log(p)$, the empirical covariance matrix is close to the true covariance matrix [139].

Similar to typical PCA, we show that the empirical center and empirical covariance matrix of the new data samples $\{\mathbf{R}_i\mathbf{y}_i\}_{i=1}^n$ (scaled by a known factor) result in accurate estimates of the original center $\overline{\mathbf{x}}$, and the underlying covariance matrix $\mathbf{C}_{true}$. For some intuition on this choice, we note $\mathbf{R}_i\mathbf{y}_i{=}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i$ represents the projection in $\mathbb{R}^p$ of $\mathbf{x}_i$ onto the column space of $\mathbf{R}_i$, which would be $\mathbf{R}_i(\mathbf{R}_i^T\mathbf{R}_i)^{-1}\mathbf{R}_i^T\mathbf{x}_i$, but we have eliminated a computationally expensive matrix inverse that typically approaches the identity here. We will provide rigorous theoretical analysis for the performance of these two estimators in terms of parameters such as m/p, n, SNR, and moments μ_k.

We will be particularly interested in applying our general distribution results to the case of very sparse measurement matrices. Achlioptas [1] first showed that, in the classic Johnson–Lindenstrauss result on pairwise distance preservation, the dense Gaussian projection matrices can be replaced with sparse projection matrices, where each entry is distributed on $\{-1, 0, 1\}$ with probabilities $\{\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\}$, achieving a three-fold speedup in processing time. Li et al. [87] then drew each entry from $\{-1, 0, 1\}$ with probabilities $\{\frac{1}{2s}, 1-\frac{1}{s}, \frac{1}{2s}\}$, achieving a more significant s-fold speedup in processing time. In this work, we refer to this second distribution as a sparse-Bernoulli distribution with sparsity parameter s. Sparse random projections have been applied in many other applications to substantially reduce computational complexity and memory requirements [103, 150, 151, 44].

Motivated by the success of these methods, we propose to recover PCs from sparse random projections of the data, in which each entry of $\{\mathbf{R}_i\}_{i=1}^n$ is drawn i.i.d. from the

sparse-Bernoulli distribution. In this case, each column of $\{\mathbf{R}_i\}_{i=1}^n$ has $\frac{p}{s}$ nonzero entries, on average. This choice has the following properties simultaneously:

- The computation cost for obtaining each projection is $\mathcal{O}(\frac{mp}{s})$ and thus the cost to acquire/access/hold in memory the data needed for the algorithm is $\mathcal{O}(\frac{mpn}{s})$. Specifically, we are interested in choosing m and s so that the compression factor $\gamma \triangleq \frac{m}{s} < 1$. In this case, our framework requires significantly less computation cost and storage space. First, the computation cost to acquire/access each data sample is $\mathcal{O}(\gamma p)$, $\gamma < 1$, in contrast to the cost for acquiring each original data sample $\mathcal{O}(p)$. This results in a substantial cost reduction for the sensing process, e.g. for streaming data. Second, once acquired, observe that the projected data samples $\{\mathbf{R}_i\mathbf{y}_i\}_{i=1}^n \in \mathbb{R}^p$ will be sparse, having at most $\mathcal{O}(\gamma p)$ nonzero entries each. This results in a significant reduction, $\mathcal{O}(\gamma pn)$ as opposed to $\mathcal{O}(pn)$, in memory/storage requirements and/or communication cost, e.g. transferring distributed data to a central processing unit.

- Given the sparse data matrix formed by $\{\mathbf{R}_i\mathbf{y}_i\}_{i=1}^n$, one can make use of efficient algorithms for performing a (partial) SVD on very large sparse matrices, such as the Lanczos algorithm [65] and [23]. In general, for a $p \times n$ matrix, the computational cost of the SVD is $\mathcal{O}(p^2 n)$. However, for large sparse matrices such as ours, the cost can be reduced to $\mathcal{O}(\gamma p^2 n)$ [89]. Furthermore, one can use very recent "gap-free" partial SVD results [10].

In the remainder of this chapter, we will characterize the accuracy of the estimated center and PCs in terms of m, p, n, SNR, moments of the distribution (which for sparse-Bernoulli will scale with s), etc. As we will see, under certain conditions on the PCs, we may choose γ as low as $\gamma \propto \frac{1}{p}$ for constant accuracy. Hence, assuming $n = \mathcal{O}(p)$ samples, the memory/storage requirements for our approach can scale with p in contrast to p^2 for standard algorithms that store the full data, and a similar factor of p savings in computation can be achieved compared

with the regular SVD. Less aggressive savings will also be available for other PC types.

2.5 Main Results

We present the main results of our work in this section, with all proofs delayed to Appendix A. Interestingly, we will see that the shape of the distribution for each entry of $\{\mathbf{R}_i\}_{i=1}^n$ plays an important role in our results. The kurtosis, defined as $\kappa \triangleq \frac{\mu_4}{\mu_2^2} - 3$, is a measure of peakedness and heaviness of tail for a distribution. It can also be thought of as a measure of non-Gaussianity, since the kurtosis of the Gaussian distribution is zero. It turns out that the distribution's kurtosis is a key factor in determining PC estimation accuracy. For sparse-Bernoulli, the kurtosis increases with increasing sparsity parameter s.

2.5.1 Mean and variance of center estimator

Theorem 2.1. *Assume that* $\{\mathbf{R}_i\}_{i=1}^n$, $\{\mathbf{x}_i\}_{i=1}^n$, $\{\mathbf{y}_i\}_{i=1}^n$, m, n, *and* μ_2 *are as defined in Section 2.3, and define the* n*-sample center estimator*

$$\widehat{\overline{\mathbf{x}}}_n = \frac{1}{m\mu_2}\frac{1}{n}\sum_{i=1}^{n}\mathbf{R}_i\mathbf{y}_i. \tag{2.2}$$

Then, the mean of the estimator $\widehat{\overline{\mathbf{x}}}_n$ *is the true center of the original data* $\overline{\mathbf{x}}$, *i.e.* $\mathbb{E}[\widehat{\overline{\mathbf{x}}}_n] = \overline{\mathbf{x}}$, *for all* n, *including the base case* n=1. *Furthermore, as* $n \to \infty$, *the estimator* $\widehat{\overline{\mathbf{x}}}_n$ *converges to the true center:* $\lim_{n\to\infty}\widehat{\overline{\mathbf{x}}}_n = \overline{\mathbf{x}}$.

Proof. See Appendix A.2. □

We see that the empirical center of $\{\mathbf{R}_i\mathbf{y}_i\}_{i=1}^n$ is a (scaled) unbiased estimator for the true center $\overline{\mathbf{x}}$. Note that this theorem does not depend on the number of projections m or sparsity parameter s, and thus does not depend on γ, as a sufficiently high number of samples will compensate for unfavorable values of these parameters. We further note that, when $n \to \infty$, there is no difference between the Gaussian, very sparse, or other choices of random projections. This is consistent with the observation that random projection

matrices consisting of i.i.d. entries must only be zero mean to preserve pairwise distances in the Johnson–Lindenstrauss theorem [87].

Theorem 2.2. *Assume that* $\{\mathbf{R}_i\}_{i=1}^n$, $\{\mathbf{x}_i\}_{i=1}^n$, $\{\mathbf{y}_i\}_{i=1}^n$, m, n, p, μ_2, h, *and SNR are as defined in Section 2.3, and kurtosis* κ *is as defined above. Then, the variance of the unbiased center estimator* $\widehat{\overline{\mathbf{x}}}_n$ *is*

$$Var\Big(\widehat{\overline{\mathbf{x}}}_n\Big) = \frac{1}{n\frac{m}{p}}\Big(h\Big(1+\frac{1}{SNR}\Big)\Big(1+\frac{m}{p}+\frac{\kappa+1}{p}\Big)+\Big(1+\frac{\kappa+1}{p}\Big)\left\|\overline{\mathbf{x}}\right\|_2^2\Big). \tag{2.3}$$

Proof. See Appendix A.3. □

We see that as the number of samples n and measurement ratio m/p increase, the variance of this estimator decreases at rate $\frac{1}{n}$ and close to $\frac{1}{m/p}$. Interestingly, the power of the signal, i.e. $h{=}\sum_{j=1}^d \sigma_j^2$, works against the accuracy of the estimator. The intuition for this is that, for the center estimation problem, it is desirable to have all the data samples close to the center, which happens for small h. For sparse random projections, we observe that the kurtosis is $\kappa{=}s-3$ and thus $\frac{\kappa+1}{p} \approx \frac{s}{p}$. Hence, variance scales with increasing sparsity, although sufficient data samples n are enough to combat this effect. Indeed, when $s > p$, the variance increases heavily since many of the random vectors are zero, and thus the corresponding projections cannot capture any information about the original data. Overall, this result shows an explicit tradeoff between reducing n or increasing s to reduce memory/computation and the variance of the resulting estimator. Finally, given this mean and variance, probabilistic error bounds can be obtained via Chebyshev, Bernstein, etc. inequalities.

2.5.2 Mean of covariance estimator

Theorem 2.3. *Assume that* $\{\mathbf{R}_i\}_{i=1}^n$, $\{\mathbf{x}_i\}_{i=1}^n$, $\{\mathbf{y}_i\}_{i=1}^n$, m, n, p, μ_2, h, ϵ, *and* $\mathbf{C}_{true}$ *are as defined in Section 2.3, and* κ *is the kurtosis. Moreover, assume that* $\{\mathbf{x}_i\}_{i=1}^n$ *are centered at* $\overline{\mathbf{x}} = \mathbf{0}$. *Define the n-sample covariance estimator*

$$\widehat{\mathbf{C}}_n = \frac{1}{(m^2+m)\,\mu_2^2}\frac{1}{n}\sum_{i=1}^n \mathbf{R}_i\mathbf{y}_i\mathbf{y}_i^T\mathbf{R}_i^T. \tag{2.4}$$

Then, for all n, the mean of this estimator is

$$\mathbf{C}_\infty \triangleq \mathbb{E}[\widehat{\mathbf{C}}_n] = \widehat{\mathbf{C}}_{true} + \mathbf{E} \tag{2.5}$$

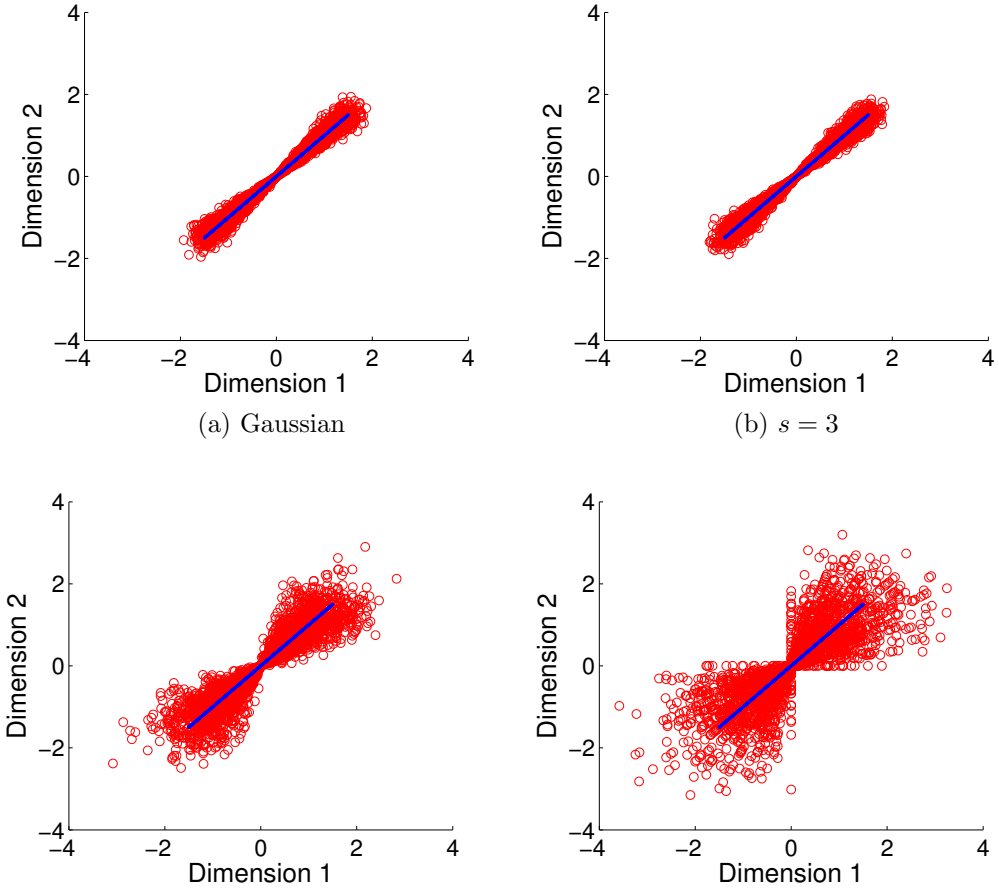
where $\widehat{\mathbf{C}}_{true} \triangleq \mathbf{C}_{true} + \alpha \mathbf{I}_{p\times p}$, $\alpha \triangleq \frac{h}{m+1} + (\frac{\kappa}{p(m+1)} + \frac{(m+p+1)}{p(m+1)})\epsilon^2$, *and* $\mathbf{E} \triangleq \frac{\kappa}{m+1}\sum_{j=1}^{d} \sigma_j^2 \, diag(\mathbf{v}_j \mathbf{v}_j^T)$, *where* $diag(\mathbf{A})$ *denotes the matrix formed by zeroing all but the diagonal entries of* $\mathbf{A}$. *Furthermore, as* $n \to \infty$, *the estimator* $\widehat{\mathbf{C}}_n$ *converges to* $\mathbf{C}_\infty$*:* $\lim_{n\to\infty} \widehat{\mathbf{C}}_n = \mathbf{C}_\infty$.

Proof. See Appendix A.4. □

We observe that the limit of the estimator $\widehat{\mathbf{C}}_n$ has two components. The first, $\widehat{\mathbf{C}}_{true}$, has the same eigenvectors with slightly perturbed eigenvalues (α tends to be very small in high dimensions) and the other, $\mathbf{E}$, is an error perturbation term. Both α and $\mathbf{E}$ scale with the kurtosis, reflecting the necessary tradeoff between increasing sparsity (decreasing memory/computation) and maintaining accuracy.

We now consider a simple example to gain some intuition for this theorem. A set of data samples $\{\mathbf{x}_i\}_{i=1}^{3000} \in \mathbb{R}^{1000}$ are generated from one PC. We also generate the random projection matrices $\{\mathbf{R}_i\}_{i=1}^{3000} \in \mathbb{R}^{1000\times 200}$ ($m/p = 0.2$) with i.i.d. entries, both for the Gaussian distribution and the sparse-Bernoulli distribution for various values of the sparsity parameter s. In Fig. 2.1, we view two dimensions (the original PC's and one other) of the data $\{\mathbf{x}_i\}_{i=1}^{3000}$ and the scaled projected data $\frac{1}{\sqrt{(m^2+m)\mu_2^2}}\{\mathbf{R}_i\mathbf{y}_i\}_{i=1}^{3000}$, represented by blue dots and red circles respectively. We see that the projected data samples are scattered somewhat into directions other than the PC's for all four cases. However, we see that the covariance of the scattered energy depends on $\mathbf{R}_i$'s distribution only through its kurtosis. For both the Gaussian and sparse-Bernoulli with $s = 3$ cases, the kurtosis is $\kappa = 0$, so the projected data cloud takes the same shape for both, with the overall amount of scattering quite small due to the low kurtosis value. As we increase the parameter s, the kurtosis $\kappa = s-3$ gets larger, and the projected data samples get more scattered into other directions. We also note the similarity of our findings to the result in [87] that the variance of the pairwise distances in Johnson–Lindenstrauss

depends also on the kurtosis of the distribution being used for random projections. Despite the perturbation, in all cases, the PC can be recovered accurately. Note also that scaling the projected data points by $1/\sqrt{(m^2+m)\,\mu_2^2}$ preserves the energy in the direction of the PC (i.e. the eigenvalue).

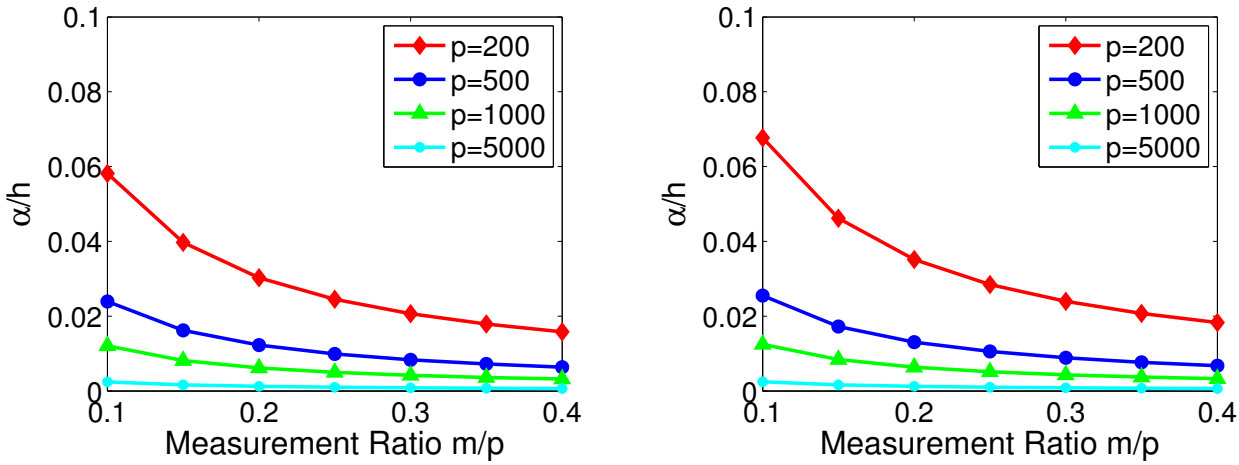


(a) Gaussian (b) $s = 3$

(c) $s = 20$ (d) $s = 50$

Figure 2.1: Accurate recovery of the PC under random projections using both Gaussian and sparse random projection matrices for various values of s. In each figure, there are $n = 3000$ data samples uniformly distributed on a line in $\mathbb{R}^{1000}$. $\{\mathbf{R}_i\}_{i=1}^n \in \mathbb{R}^{p\times m}$, $m/p = 0.2$, are generated with i.i.d. entries drawn from (a) $\mathcal{N}(0,1)$ and (b,c,d) the sparse-Bernoulli distribution for $s = 3, 20, 50$. In each figure, we view two dimensions (the original PC's and one other) of the data $\{\mathbf{x}_i\}_{i=1}^n$ (blue dots) and the scaled projected data $(1/((m^2+m)\mu_2^2)^{1/2})\{\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\}_{i=1}^n$ (red circles). We see that, in all cases, the projected data are symmetrically distributed around the PC, and the inner product magnitude between the PC estimated from the projected data and the true PC is at least 0.99.

In Theorem 2.3, we see that $\mathbf{C}_{true}$ and $\widehat{\mathbf{C}}_{true}$ have the same set of eigenvectors with the eigenvalues of $\mathbf{C}_{true}$ increased by $\alpha = h\{\frac{1}{m+1}+(\frac{\kappa}{p(m+1)}+\frac{1}{p}+\frac{1}{m+1})\frac{1}{\text{SNR}}\}$. Thus, α is a decreasing function of p, m/p and SNR, and in particular goes to 0 as $p \to \infty$ for constant measurement ratio m/p. This is illustrated in Fig. 2.2. Thus, surprisingly, in the high-dimensional regime, the amount of perturbation of eigenvalues becomes increasingly negligible even for small measurement ratios.



(a) (b)

Figure 2.2: Variation of the parameter $\frac{\alpha}{h}$ for (a) $\kappa = 0$ and (b) $\kappa = 200$, for varying p and measurement ratio m/p, and fixed SNR $= 5$.

Now, let's examine the error matrix $\mathbf{E}$. We observe that $\mathbf{E}$ can be viewed as representing a bias of the estimated PCs towards the nearest canonical basis vectors when the kurtosis is high, and this bias is visible in Fig. 2.1 as well. It stems from anisotropy in the distribution for $\mathbf{R}_i$ when this is non-Gaussian (note κ=0, and thus $\mathbf{E}$=$\mathbf{0}$, for the Gaussian case). In later sections, we will use the 2-norm (spectral norm) of $\mathbf{E}$, $\|\mathbf{E}\|_2$, to bound the angle between the estimated and true PCs. Indeed, we find, for constant $\delta \triangleq \frac{\|\mathbf{E}\|_2}{h}$, the same angular PC estimation error is achieved.

In the next section, we study $\|\mathbf{E}\|_2$, leading to useful observations, for several types of PCs. We see that our method is effective for many different types of PCs, although the

greatest savings in memory/computation are achieved for smooth PCs.

2.5.3 Discussion of covariance estimator error by PC types

Before we begin, we define the mutual coherence μ_{max} between the PCs and the canonical basis $\{\mathbf{e}_j\}_{j=1}^p$ as $\mu_{max} \triangleq \max_{i=1,\ldots,d,j=1,\ldots,p} |\langle \mathbf{v}_i, \mathbf{e}_j \rangle|$ [50] and will also define μ_{min} as $\mu_{min} \triangleq \min_{i=1,\ldots,d} \max_{j=1,\ldots,p} |\langle \mathbf{v}_i, \mathbf{e}_j \rangle|$. Note that μ_{min} is a measure of closeness of the PCs with the canonical basis vectors. In fact, a value of μ_{min} close to 1 represents the case when all of the PCs are (approximately) sparse.

2.5.3.1 Smooth PCs

It has frequently been observed that sparse-Bernoulli random projections are most effective on vectors that are "smooth" [7], meaning that their maximum entry is of size $\mathcal{O}(\frac{1}{\sqrt{p}})$. Large images, videos, and other natural signals with distributed energy are obvious examples of this type. (Other signals are often preconditioned to be smooth via multiplication with a Hadamard conditioning matrix, e.g. as in [44].) Note that $\mathbf{E}$ is a diagonal matrix whose k^{th} diagonal element is $\frac{\kappa}{m+1} \sum_{j=1}^d \sigma_j^2 v_{j,k}^2$, where $\mathbf{v}_j = [v_{j,1}, \ldots, v_{j,p}]^T$. Hence, we may easily observe that $\|\mathbf{E}\|_2 \leq \frac{\kappa}{m+1} \mu_{max}^2 h$, or $\delta \leq \frac{\kappa}{m+1} \mu_{max}^2$, and we also note $\frac{\kappa}{m+1} < \frac{1}{\gamma}$ for sparse random projections. As we will see in Section 2.5.5, we will want to keep δ small enough to guarantee a certain fixed angular error θ_0. In fact, this can be satisfied by requiring $\gamma \geq C(\theta_0)\mu_{max}^2$, where $C(\theta_0)$ is a constant depending on the error θ_0. Hence, for smooth PCs, we need only have $\gamma \propto \frac{1}{p}$, reducing memory and computation by a rather remarkable factor of p.

2.5.3.2 All sparse PCs

At times, we may be interested in cases when the PCs are sparse. For example, datasets with sparse PCs are often considered particularly amenable to interpretation of the data. We consider this case to be of lesser interest for our algorithm than that of smooth PCs,

since there are already many existing algorithms specifically tailored for recovering sparse PCs efficiently [80, 152, 105]. Nevertheless, it is interesting to see what happens in this case.

Here, the mutual coherence μ_{max} may be close to 1. Hence, in contrast to the case of smooth PCs, the amount of perturbation $\|\mathbf{E}\|_2 \leq \frac{\kappa}{m+1}\mu_{max}^2 h \leq \frac{1}{\gamma}\mu_{max}^2 h$ is relatively high for small values of the compression factor, e.g. when $\gamma \propto \frac{1}{p}$. Fortunately, however, the perturbation $\mathbf{E}$ tends to align with the canonical basis. For sparse PCs then, which are also aligned with the canonical basis, the effect, though large, is along the PCs, and thus not harmful to PC estimation accuracy.

To see this, we first consider an extreme case when all d PCs are chosen from among the canonical basis vectors. In this case, $\text{diag}(\mathbf{v}_j\mathbf{v}_j^T) = \mathbf{v}_j\mathbf{v}_j^T$ exactly, j=1, . . . , d. Hence, the perturbation term $\mathbf{E} = \frac{\kappa}{m+1}\sum_{j=1}^{d}\sigma_j^2\text{diag}(\mathbf{v}_j\mathbf{v}_j^T) = \frac{\kappa}{m+1}\sum_{j=1}^{d}\sigma_j^2\mathbf{v}_j\mathbf{v}_j^T = \frac{\kappa}{m+1}\mathbf{C}_{true}$ and the limit of $\widehat{\mathbf{C}}_n$ is

$$\mathbf{C}_\infty = \mathbf{C}_{true} + \alpha\mathbf{I}_{p\times p} + \mathbf{E} = \Big(1 + \frac{\kappa}{m+1}\Big)\mathbf{C}_{true} + \alpha\mathbf{I}_{p\times p}. \tag{2.6}$$

Thus, the original PCs can be recovered accurately without any perturbation because the perturbation term $\mathbf{E}$ aligns exactly with the true PCs. However, the corresponding eigenvalues will be increased heavily by the known factor $1 + \frac{\kappa}{m+1} + \alpha \approx 1 + \frac{1}{\gamma}$, given that α is small in the high-dimensional regime. So we see that sparse PCs do not cause any serious issue here, apart from possible scaling of the eigenvalues.

Before continuing, we provide a simulated toy example to validate the analysis above. We synthetically generate 25,000 data samples in $\mathbb{R}^{500}$ distributed along one PC which is the first element of the canonical basis, i.e. $\mathbf{v} = \mathbf{e}_1 = [1, 0, \ldots, 0]^T$. We fix the measurement ratio $m/p = 0.3$ and estimate the PC and its corresponding eigenvalue for various compression factors γ. Fig. 2.3(a) shows the accuracy for the estimated PC. We see that our approach works well in this situation and the PC is estimated accurately, which is consistent with the analysis.

Now, armed with this intuition, let's reconsider the original problem where the PCs

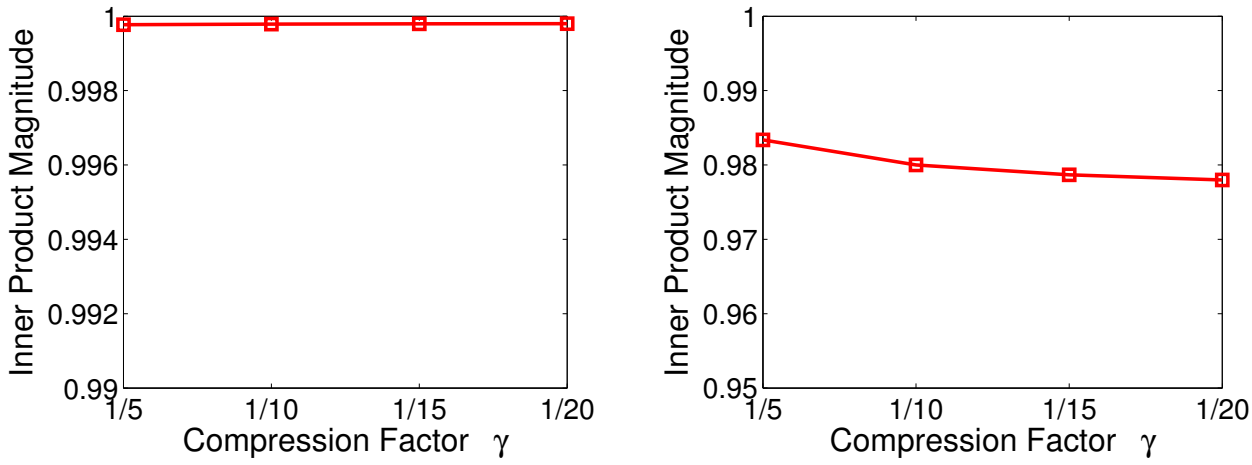


Figure 2.3: PC estimation results for synthetic data. Plot of magnitude of the inner product between the estimated and true PC for (a) $\mathbf{v} = \mathbf{e}_1$ and (b) $\mathbf{v} = [100, 20, 10, 0, \ldots, 0]^T$, normalized to have unit norm. Notice that the PC is well-estimated in both cases.

are sparse. Based on the previous discussion, it is straightforward to re-write $\mathbf{C}_\infty$ as

$$\mathbf{C}_\infty = \Big(1 + \frac{\kappa}{m+1}\Big)\mathbf{C}_{true} + \alpha \mathbf{I}_{p\times p} + \mathbf{E}_0 \tag{2.7}$$

where $\mathbf{E}_0 \triangleq \frac{\kappa}{m+1}\sum_{j=1}^{d}\sigma_j^2(\mathrm{diag}(\mathbf{v}_j\mathbf{v}_j^T) - \mathbf{v}_j\mathbf{v}_j^T)$ is a new error perturbation term, created by removing the part of the perturbation that aligns with the true sparse PCs and is not harmful.

We wish to upper bound the amount of potentially harmful perturbation error $\|\mathbf{E}_0\|_2$. We start by bounding $\left\|\mathbf{v}\mathbf{v}^T - \mathrm{diag}\left(\mathbf{v}\mathbf{v}^T\right)\right\|_2$ for a fixed vector $\mathbf{v}$ with unit ℓ_2-norm:

$$\begin{aligned}
&\left\|\mathbf{v}\mathbf{v}^T - \mathrm{diag}\left(\mathbf{v}\mathbf{v}^T\right)\right\|_2 \\
&\leq \left\|\mathbf{v}\mathbf{v}^T - \mathrm{diag}\left(\mathbf{v}\mathbf{v}^T\right)\right\|_F \\
&= \Big(\sum_{k\neq j} v_k^2 v_j^2\Big)^{1/2} = \Big(\sum_{k,j} v_k^2 v_j^2 - \sum_{k=1}^{p} v_k^4\Big)^{1/2} \\
&\overset{(i)}{=} \Big(1 - \sum_{k=1}^{p} v_k^4\Big)^{1/2} \overset{(ii)}{=} \Big(1 - \frac{\sum_{k=1}^{p} v_k^4}{(\sum_{k=1}^{p} v_k^2)^2}\Big)^{1/2}
\end{aligned} \tag{2.8}$$

where (i) and (ii) both follow from the fact that $\|\mathbf{v}\|_2^2 = 1$.

The upper bound in (2.8) gives us important information about the amount of perturbation. The term $\frac{\sum_{k=1}^{p} v_k^4}{\left(\sum_{k=1}^{p} v_k^2\right)^2}$ is close to 1 (i.e. the amount of perturbation is very small) when a few entries of the PC are nonzero. Moreover, this term gets closer to 1 when the nonzero entries have a fast rate of decay. For example, let us consider the extreme case of coherence 1 between $\mathbf{v}$ and the canonical basis. In this case, it is easy to see that the term $\frac{\sum_{k=1}^{p} v_k^4}{\left(\sum_{k=1}^{p} v_k^2\right)^2}$ is exactly one and thus the amount of harmful perturbation is zero. This is consistent with our prior observation.

Therefore, a useful upper bound for the 2-norm of the perturbation term $\mathbf{E}_0$ can be found by considering

$$\begin{aligned} \left\|\mathbf{v}_j\mathbf{v}_j^T - \operatorname{diag}\left(\mathbf{v}_j\mathbf{v}_j^T\right)\right\|_2 &\leq \left(1 - \sum_{k=1}^{p} v_{j,k}^4\right)^{1/2} \\ &\leq \left(1 - \left(\max_{k=1,\ldots,p} |\langle \mathbf{v}_j, \mathbf{e}_k\rangle|\right)^4\right)^{1/2} \end{aligned} \tag{2.9}$$

and this results in

$$\begin{aligned} \|\mathbf{E}_0\|_2 &\leq \frac{\kappa}{m+1} \sum_{j=1}^{d} \sigma_j^2 \left\|\mathbf{v}_j\mathbf{v}_j^T - \operatorname{diag}\left(\mathbf{v}_j\mathbf{v}_j^T\right)\right\|_2 \\ &\leq \frac{\kappa}{m+1} h\sqrt{1-\mu_{min}^4}. \end{aligned} \tag{2.10}$$

Thus, we find that sparse PCs can still be recovered very well here, although the eigenvalues may be heavily scaled by the known factor $1+\frac{\kappa}{m+1}$. Allowing this, and taking $\mathbf{E}_0$ as the error term instead of $\mathbf{E}$, we see that we can let $\gamma \propto \sqrt{1-\mu_{min}^4}$ to maintain constant δ and hence constant PC estimation error.

To validate this analysis, we consider the same simulated toy example described in the previous case. However, we now consider the PC $\mathbf{v} = [100, 20, 10, 0, \ldots, 0]^T$, normalized to have unit ℓ_2-norm. Fig. 2.3(b) shows the accuracy for the estimated PC. We see that our approach is less accurate than it was for the canonical basis vector, but still works quite well.

As a final note, our result for sparse PCs may seem surprising compared with the literature on the Johnson–Lindenstrauss (JL) theorem, where sparse-Bernoulli random projections are frequently used. For example, the authors in [7] observed that sparse data

samples (which they call "bad inputs"), can cause high variance in the JL transform when using sparse-Bernoulli random projections. Intuitively, this is due to the fact that sparse projections can capture information about the structure of the original data only when the nonzero entries of the random projections are aligned with nonzero coordinates of the data. Thus, when the data is very sparse, the probability of the nonzero entries of the random projections failing to overlap with the few nonzero coordinates of the data, and hence receiving zero information from a projection, is high. The authors in [7] suggest preconditioning the data using a randomized Hadamard matrix in the JL transform as a fix for this problem.

2.5.3.3 Neither sparse nor smooth PCs

In this case, we may follow the model of the low-coherence case, and still obtain the result that γ may scale with μ_{max}^2 to achieve fixed accuracy. However, μ_{max}^2 is not as small as in the low-coherence case, and the memory and computation savings will thus not be as aggressive, although they may still be substantial.

2.5.3.4 Mixture of PC types

In this case, we may split $\mathbf{E}$ into two error matrices, associated with each of the sparse and non-sparse PCs. Recovery of the d-dimensional PC subspace still performs well here. However, if the eigenvalues do not decay sufficiently fast, scaling of the eigenvalues for the sparse PCs may reorder the individual components. To be precise, let us define $\mathcal{I}_{low}$ and $\mathcal{I}_{high}$ as the set of indices of PCs with low/medium and high coherence respectively, then we

see that:

$$\begin{aligned}
\mathbf{C}_\infty &= \sum_{j=1}^{d} \sigma_j^2 \Big(\mathbf{v}_j \mathbf{v}_j^T + \frac{\kappa}{m+1} \mathrm{diag}\left(\mathbf{v}_j \mathbf{v}_j^T\right) \Big) + \alpha \mathbf{I}_{p\times p} \\
&= \alpha \mathbf{I}_{p\times p} + \sum_{j\in\mathcal{I}_{low}} \sigma_j^2 \Big(\mathbf{v}_j \mathbf{v}_j^T + \frac{\kappa}{m+1} \mathrm{diag}\left(\mathbf{v}_j \mathbf{v}_j^T\right) \Big) \\
&+ \sum_{j\in\mathcal{I}_{high}} \sigma_j^2 \Big(\Big(1 + \frac{\kappa}{m+1}\Big) \mathbf{v}_j \mathbf{v}_j^T + \frac{\kappa}{m+1} \Big(\mathrm{diag}\left(\mathbf{v}_j \mathbf{v}_j^T\right) - \mathbf{v}_j \mathbf{v}_j^T \Big) \Big) \\
&= \alpha \mathbf{I}_{p\times p} + \sum_{j\in\mathcal{I}_{low}} \sigma_j^2 \mathbf{v}_j \mathbf{v}_j^T + \sum_{j\in\mathcal{I}_{high}} \underbrace{\Big(1 + \frac{\kappa}{m+1}\Big) \sigma_j^2}_{\text{new eigenvalues}} \mathbf{v}_j \mathbf{v}_j^T \\
&+ \underbrace{\frac{\kappa}{m+1} \Big(\sum_{j\in\mathcal{I}_{low}} \sigma_j^2 \mathrm{diag}\left(\mathbf{v}_j \mathbf{v}_j^T\right) + \sum_{j\in\mathcal{I}_{high}} \sigma_j^2 \left(\mathrm{diag}\left(\mathbf{v}_j \mathbf{v}_j^T\right) - \mathbf{v}_j \mathbf{v}_j^T \right) \Big)}_{\text{2 error terms}}.
\end{aligned}$$

The two error terms remain small, but for large γ, scaling of the eigenvalues in $\mathcal{I}_{high}$ may reorder the original eigenvalues, and it may become possible only to recover the d-dimensional subspace containing all the PCs, not the original ordering.

2.5.4 Deviation of covariance estimator from its mean

In Section 2.5.2, we showed that the n-sample covariance estimator $\widehat{\mathbf{C}}_n$ converges to $\mathbf{C}_\infty$ as n tends to infinity. However, it is important to study how fast our covariance estimator converges to its mean $\mathbf{C}_\infty$ as n grows. This allows us to understand the effect of the parameters of our model such as the compression factor γ on the convergence rate.

Theorem 2.4. *Assume that* $\{\mathbf{R}_i\}_{i=1}^n$, $\{\mathbf{x}_i\}_{i=1}^n$, $\{\mathbf{y}_i\}_{i=1}^n$, m, n, p, μ_k, h, *and SNR are as defined in Section 2.3. Consider the covariance matrix estimator* $\widehat{\mathbf{C}}_n = \frac{1}{m(m+1)\mu_2^2} \frac{1}{n} \sum_{i=1}^n \mathbf{R}_i \mathbf{y}_i \mathbf{y}_i^T \mathbf{R}_i^T$. *Then, the deviation of our n-sample estimator from its mean value is upper bounded*

$$\mathbb{E}\Big[\Big\| \widehat{\mathbf{C}}_n - \mathbf{C}_\infty \Big\|_F^2 \Big] \leq \frac{1}{n} \tau h^2 \tag{2.11}$$

where

$$\tau = \xi \Big\{ \Big(1 + \frac{1}{SNR}\Big)^2 + \Big(\beta \frac{\widetilde{h}}{h^2} + \frac{4}{p} \frac{1}{SNR} + \frac{2}{p} \Big(\frac{1}{SNR} \Big)^2 \Big) \Big\},$$

where $\widetilde{h} \triangleq \sum_{j=1}^{d} \sigma_j^4$, $\beta \triangleq \mathbb{E}[w^4] - 1$, *and* $\xi = \max(\xi_1, \xi_2)$, *where*

$$\xi_1 \leq \frac{\mu_8/\mu_2^4}{m^3} + 2\frac{\mu_6/\mu_2^3}{m^2}\Big(2 + \frac{1}{m/p}\Big) + \frac{\big(\mu_4/\mu_2^2\big)^2}{m^2}\Big(3 + \frac{1}{m/p}\Big) + \frac{\mu_4/\mu_2^2}{m}\Big(3 + \frac{1}{m/p}\Big)^2 + \Big(1 + \frac{1}{m/p}\Big)^2 + \frac{3}{p}\Big(\frac{1}{m/p}\Big)^2 \tag{2.12}$$

and

$$\xi_2 \leq \frac{\mu_6/\mu_2^3}{m^2}\Big(\frac{6}{m}\Big) + \frac{\big(\mu_4/\mu_2^2\big)^2}{m^2}\Big(1 + \frac{5}{m}\Big) + \frac{\mu_4/\mu_2^2}{m}\Big(2 + \frac{13}{m}\Big)\Big(2 + \frac{1}{m/p}\Big) + \Big(1 + \frac{1}{m/p}\Big)^2 + \frac{4}{m}\Big(2 + \frac{1}{m/p}\Big)^2. \tag{2.13}$$

Proof. See Appendix A.5. □

Note that ξ has various terms that scale with $\frac{1}{p}$, $\frac{1}{m/p}$, and the higher order moments μ_8/μ_2^4, μ_6/μ_2^3, and μ_4/μ_2^2.

Theorem 2.4 leads to several important observations. We see that as the number of data samples n increases, the variance decreases at rate $\frac{1}{n}$, converging quickly to the limit. Moreover, the variance of our estimator is a decreasing function of the measurement ratio m/p and SNR. We further note that the parameter ξ gives us important information about the effect of the tails of the distribution on the convergence rate of the covariance estimator. More precisely, for sparse random projections, we see that $\frac{\mu_8/\mu_2^4}{m^3} = (\frac{s}{m})^3 = \frac{1}{\gamma^3}$, $\frac{\mu_6/\mu_2^3}{m^2} = \frac{(\mu_4/\mu_2^2)^2}{m^2} = \frac{1}{\gamma^2}$, and $\frac{\mu_4/\mu_2^2}{m} = \frac{1}{\gamma}$. Hence, for a fixed n, decreasing the compression factor γ leads to an increase of the variance and a loss in accuracy. This is as we would expect since there is an inherent tradeoff between saving computation and memory and the accuracy. However, characterizing this tradeoff allows γ to be chosen in an informed way for large datasets.

2.5.5 Memory, computation, and PC accuracy tradeoffs

We now use the covariance estimator results to bound the error of its eigenvalues and eigenvectors, using related results from matrix perturbation theory.

First, note that using the variance of our estimator (2.11) in the Chebyshev inequality yields $\left\|\widehat{\mathbf{C}}_n - \mathbf{C}_\infty\right\|_F \leq \varepsilon h$, with probability at least $1 - \frac{1}{n\varepsilon^2}\tau$. Hence,

$$\begin{aligned}\left\|\widehat{\mathbf{C}}_n - \widehat{\mathbf{C}}_{true}\right\|_2 &\leq \left\|\widehat{\mathbf{C}}_n - \mathbf{C}_\infty\right\|_2 + \left\|\mathbf{C}_\infty - \widehat{\mathbf{C}}_{true}\right\|_2 \\ &\leq \left\|\widehat{\mathbf{C}}_n - \mathbf{C}_\infty\right\|_F + \|\mathbf{E}\|_2 \leq (\varepsilon + \delta)\, h \end{aligned} \tag{2.14}$$

with probability at least $1 - \frac{1}{n\varepsilon^2}\tau$. In fact, (2.14) can be used to characterize tradeoffs between memory, computation, and PC estimation accuracy (as an angle between estimated subspaces) in terms of our parameters n, m/p, etc.

For illustrative purposes, we start by analyzing the case of a single PC and use the following Lemma. In the following, $\lambda(\mathbf{A})$ and $\lambda_i(\mathbf{A})$ denote the set of all eigenvalues and the i^{th} eigenvalue of $\mathbf{A}$, respectively.

Lemma 2.1. *([69, 43]) Suppose* $\mathbf{A}$ *is a real symmetric matrix and* $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{E}$ *is the perturbed matrix. Assume that* $(\widetilde{\lambda}, \widetilde{\mathbf{v}})$ *is an exact eigenpair of* $\widetilde{\mathbf{A}}$ *where* $\|\widetilde{\mathbf{v}}\|_2 = 1$*. Then*

(a) $\left|\widetilde{\lambda} - \lambda\right| \leq \|\mathbf{E}\|_2$ *for some eigenvalue* λ *of* $\mathbf{A}$*.*

(b) Let λ *be the closest eigenvalue of* $\mathbf{A}$ *to* $\widetilde{\lambda}$ *and* $\mathbf{v}$ *be its associated eigenvector with* $\|\mathbf{v}\|_2 = 1$*, and let* $\eta = \min_{\lambda_0 \in \lambda(\mathbf{A}), \lambda_0 \neq \lambda} \left|\widetilde{\lambda} - \lambda_0\right|$ *. If* $\eta > 0$*, then*

$$\sin \angle (\widetilde{\mathbf{v}}, \mathbf{v}) \leq \frac{\|\mathbf{E}\|_2}{\eta} \tag{2.15}$$

where $\angle (\widetilde{\mathbf{v}}, \mathbf{v})$ *denotes the canonical angle between the two eigenvectors.*

We will use this Lemma to bound the angle between the PC estimate from $\widehat{\mathbf{C}}_n$ and the true PC in the single PC case. Since $\mathbf{C}_{true}$ has only one eigenpair $(\sigma^2, \mathbf{v})$ with nonzero eigenvalue, $\widehat{\mathbf{C}}_{true}$ has an eigenpair $(\sigma^2 + \alpha, \mathbf{v})$ and $\lambda_i(\widehat{\mathbf{C}}_{true}) = \alpha$, $i = 2, \ldots, p$. From Lemma 2.1, we see that the largest eigenvalue of $\widehat{\mathbf{C}}_n$ satisfies $\left|\lambda_1(\widehat{\mathbf{C}}_n) - (\sigma^2 + \alpha)\right| \leq (\varepsilon + \delta)\, \sigma^2$. We find the parameter η:

$$\begin{aligned}\eta &= \min_{i=2,\ldots,p} \left|\lambda_1\left(\widehat{\mathbf{C}}_n\right) - \lambda_i\left(\widehat{\mathbf{C}}_{true}\right)\right| = \left|\lambda_1\left(\widehat{\mathbf{C}}_n\right) - \alpha\right| \\ &\geq \sigma^2 - (\varepsilon + \delta)\, \sigma^2 = (1 - (\varepsilon + \delta))\, \sigma^2 \end{aligned} \tag{2.16}$$

for appropriate values of δ and ε such that $(\varepsilon + \delta) < 1$. We then get the following tradeoff between the accuracy of the estimated eigenvector and the parameters of our model:

$$\sin \angle (\widetilde{\mathbf{v}}, \mathbf{v}) \leq \frac{(\varepsilon + \delta)}{1 - (\varepsilon + \delta)} \tag{2.17}$$

with probability at least $1 - \frac{1}{n\varepsilon^2}\tau$. This equation allows us to characterize the statistical tradeoff between the sparsity parameter s and the accuracy of the estimated PC.

For the more general case of $d' \leq d$ PCs, we can follow the same approach to obtain the following theorem.

Theorem 2.5. *Let the true covariance matrix $\mathbf{C}_{true}$, our proposed n-sample covariance estimator $\widehat{\mathbf{C}}_n$, δ, h, n, d, τ, and γ be as defined previously. Consider the two d'-dimensional subspaces spanned by the top d', for $d' < d$, eigenvectors (i.e. those corresponding to maximum eigenvalues) of $\mathbf{C}_{true}$ and of $\widehat{\mathbf{C}}_n$ respectively. Then, for any $d' < d$, the maximum canonical angle θ_{max} between these two subspaces satisfies*

$$\sin \theta_{max} \leq \frac{(\varepsilon + \delta)}{\frac{\left(\sigma^2_{d'} - \sigma^2_{d'+1}\right)}{h} - (\varepsilon + \delta)} \tag{2.18}$$

with probability at least $1 - \frac{1}{n\varepsilon^2}\tau$, and for appropriate values of m, s, *etc. such that δ satisfies $(\varepsilon + \delta) < (\sigma^2_{d'} - \sigma^2_{d'+1})/h$. Moreover, the maximum canonical angle between the subspaces spanned by all d estimated and true PCs satisfies*

$$\sin \theta_{max} \leq \frac{(\varepsilon + \delta)}{\frac{\sigma^2_d}{h} - (\varepsilon + \delta)} \tag{2.19}$$

with probability at least $1 - \frac{1}{n\varepsilon^2}\tau$, and for $(\varepsilon + \delta) < \frac{\sigma^2_d}{h}$.

Proof. See Appendix A.6. □

Hence, to ensure, with probability $1 - \frac{1}{n\varepsilon^2}\tau$, a fixed maximum angular error θ_0 when estimating the top d' PCs, i.e. to ensure $\sin \theta_{max} \leq \sin \theta_0$, we should choose γ so that $\delta \leq \frac{\sigma^2_{d'} - \sigma^2_{d'+1}}{h} \frac{\sin \theta_0}{1 + \sin \theta_0} - \varepsilon$. For smooth PCs, we may satisfy this by choosing $\gamma \geq C(\theta_0, \varepsilon)\mu^2_{max}$

for $C(\theta_0, \varepsilon) \triangleq 1/(\frac{\sigma^2_{d'} - \sigma^2_{d'+1}}{h} \frac{\sin\theta_0}{1+\sin\theta_0} - \varepsilon)$. For $d' = d$, we obtain the same result with $\frac{\sigma^2_{d'} - \sigma^2_{d'+1}}{h}$ replaced by $\frac{\sigma^2_d}{h}$.

Thus, for large enough n, we may choose $\gamma \propto \frac{1}{p}$ to achieve a fixed accuracy with very high probability. Hence, the memory/storage requirements of our method can scale with p in contrast to standard algorithms that scale with p^2, while the computational complexity of SVD can scale with p^2 as opposed to p^3. As we discussed, although the smooth case is of special interest, less aggressive, but still substantial, savings are also available for other PC types.

2.6 Experimental Results

In this section, we examine the tradeoffs between memory, computation, and accuracy for the sparse random projections approach on both synthetic and real-world datasets. First, we synthetically generate samples $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^p$ distributed along one PC with $\sigma = 20$. Each entry of the center and PC is drawn from the uniform distribution on $[0, 20)$ and $[0, 1)$, respectively. The PC is then normalized to have unit ℓ_2-norm. We consider a relatively noisy situation with SNR $= 1$. We then estimate the center of the original data from the sparse random projections, where $m/p = 0.2$, for varying n and compression factors γ. Our results are averaged over 10 independent trials. Fig. 2.4(a) shows the accuracy for the estimated center, where the error is the distance between the estimated and the true center normalized by the true center's norm. As expected, when n or dimension p increase, the compression factor γ can be tuned to achieve a substantial reduction of storage space while obtaining accurate estimates. This is desirable for high-dimensional data stream processing.

We then fix $n = 2p$, and plot the inner product magnitude between the estimated and true PC in Fig. 2.4(b) and the computation time in Fig. 2.4(d) for varying γ. We observe that, despite saving nearly two orders of magnitude in computation time and also in memory (note $\gamma = \frac{1}{50}, \frac{1}{100}, \frac{1}{200}$) compared to PCA on the full data, the PC is well-estimated.

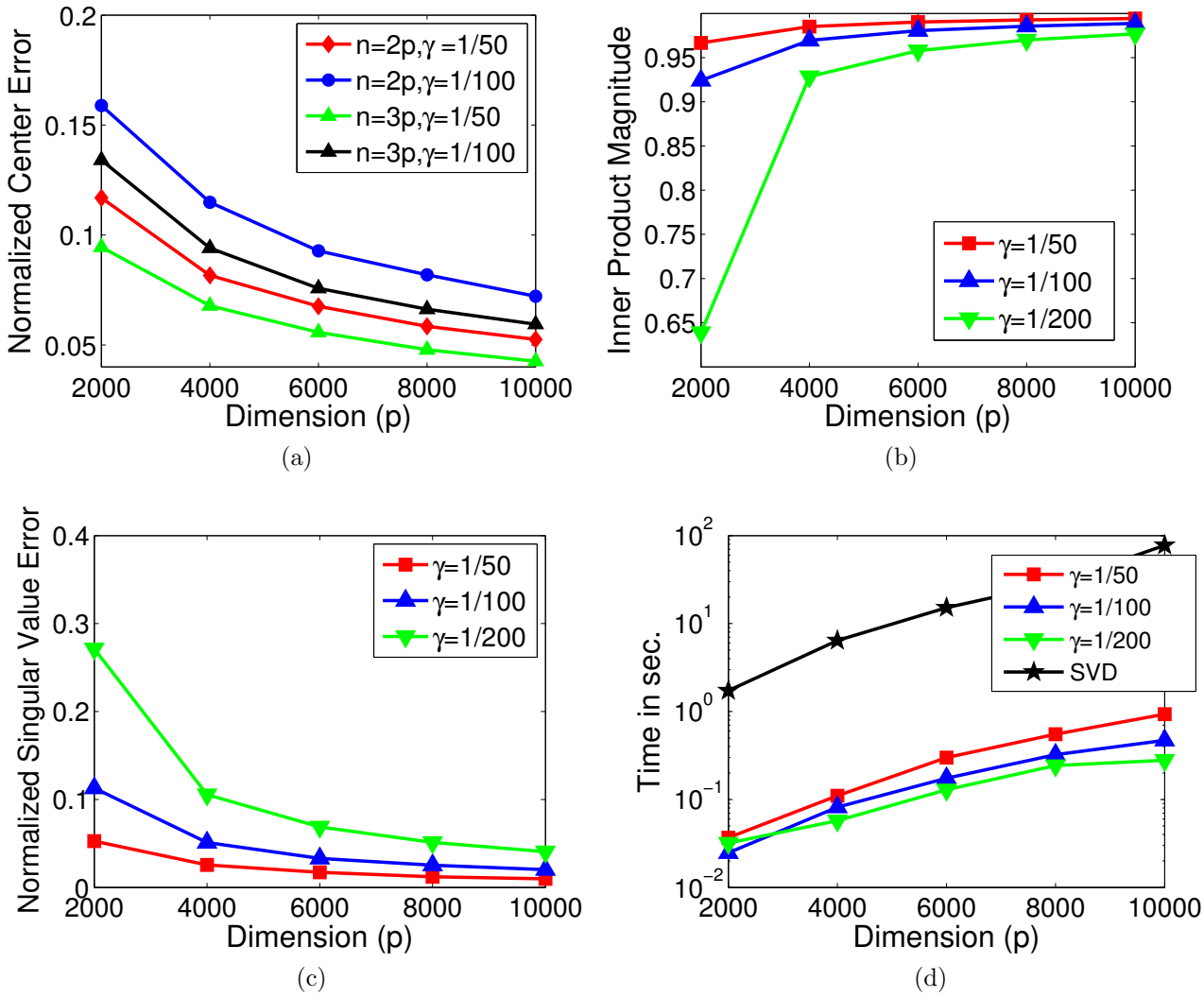


Figure 2.4: Results for synthetic data: (a) normalized estimation error for the center for varying n and γ, (b) magnitude of the inner product between the estimated and true PC for varying γ, (c) normalized estimation error for σ for varying γ, and (d) computation time to perform the SVD for the original vs. randomly projected data for varying γ.

Moreover, the approach remains increasingly effective for higher dimensions, which is of crucial importance for modern data processing applications. We further note that, as the dimension increases, we can decrease the compression factor γ while still achieving a desired performance. For example, $\gamma = \frac{1}{100}$ for $p = 4 \times 10^3$ and $\gamma = \frac{1}{200}$ for $p = 10^4$ have almost the same accuracy. This is consistent with the observation $\gamma \propto \frac{1}{p}$ from before.

We also plot the estimation error for the singular value σ in Fig. 2.4(c). The error is the distance between the singular value obtained by performing SVD on $\{\mathbf{R}_i\mathbf{y}_i\}_{i=1}^n$ and on the original data $\{\mathbf{x}_i\}_{i=1}^n$, normalized by the latter value.

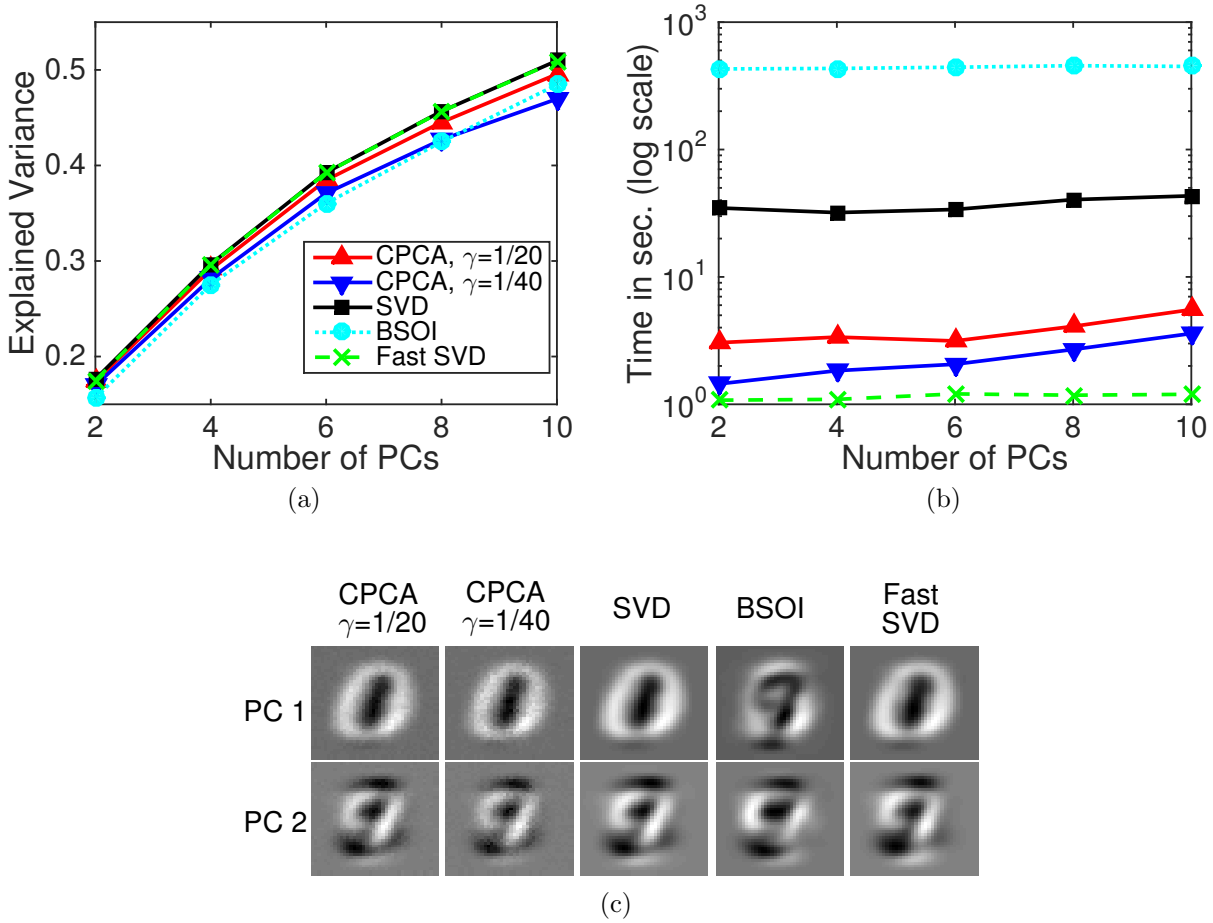


Figure 2.5: Results for the MNIST dataset. Our proposed approach (CPCA) is compared with three methods: (1) MATLAB's svds on the full original data, (2) BSOI [100], and (3) fast randomized SVD [67]. Plot of (a) performance accuracy based on the explained variance, (b) computation time for performing SVD, and (c) visual comparison of the first two estimated PCs. Our approach performs as well as SVD on the original data and fast randomized SVD and outperforms BSOI with significantly less computation time. At the same time, our method saves an order of magnitude in memory over original and fast randomized SVD.

Finally, we consider the MNIST dataset to see a real-world application outside the spiked covariance model. This dataset contains 70,000 samples of handwritten digits, which we have resized to 40×40 pixels. Hence, we have 70,000 samples in $\mathbb{R}^{1600}$. To evaluate

the performance of our method, we use the explained variance described in [100]. Given estimates of d PCs $\widetilde{\mathbf{V}} \in \mathbb{R}^{p \times d}$ and the data matrix $\mathbf{X}$, the fraction of explained variance is defined as $tr(\widetilde{\mathbf{V}}^T \mathbf{X}\mathbf{X}^T \widetilde{\mathbf{V}})/tr(\mathbf{X}\mathbf{X}^T)$. We compare the performance of our approach with three methods: (1) performing SVD (using MATLAB svds) on the original data that are fully acquired and stored, (2) the online algorithm Block-Stochastic Orthogonal Iteration (BSOI) [100], where the data samples are fully acquired but not stored, and (3) fast randomized SVD [67]. We show the results in Fig. 2.5 for the measurement ratio $m/p = 0.1$.

In terms of accuracy, our approach performs about as well as both SVD on the original data and fast randomized SVD, and has better performance compared to BSOI. Moreover, the sparse random projections result in a significant reduction of computational complexity, with one order and two orders of magnitude speedup compared to the original SVD and BSOI, respectively. We can see that fast randomized SVD is actually somewhat faster than our method, which is as we would expect, since it is designed specifically for low-computational complexity. However, fast randomized SVD is a full data method, meaning that it is assumed that the full data is available for computation and does not require time or cost to access. In terms of memory requirements, 854 MB is needed to store the original data, the amount needed for original and fast randomized SVD. However, the required memory for our framework is 83 MB for $\gamma = \frac{1}{20}$ and 42 MB for $\gamma = \frac{1}{40}$. Our approach thus performs approximately as well in similar computation time while also allowing a reduction in memory (or data access or data communication costs) by a factor of γ, in this case $\frac{1}{20}$ and $\frac{1}{40}$. This can be a significant advantage in the case where data is stored in a large database system or distributed network.

We also show a visual comparison of the first two estimated PCs in Fig. 2.5(c). We can see that the estimated PCs using our proposed approach are almost identical to those obtained by SVD on the original data.

This example indicates that our approach results in a significant simultaneous reduction of memory and/or computational cost with little loss in accuracy.

2.7 Distribution-Free Analysis

2.7.1 Problem formulation

Consider a collection of data samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^p$ and let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ be a matrix whose i-th column is the data sample $\mathbf{x}_i$. In this section, we do not make any distributional assumptions on the data and our goal is to recover the sample covariance matrix $\mathbf{C}_n$ from a single pass over the data

$$\mathbf{C}_n = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T = \frac{1}{n} \mathbf{X}\mathbf{X}^T. \tag{2.20}$$

We aim to recover $\mathbf{C}_n$ from compressive measurements of the data. This can be viewed as forming a random matrix $\mathbf{R}_i \in \mathbb{R}^{p \times m}$ with $m < p$ for $\mathbf{x}_i$, $i = 1, \ldots, n$. The entries of $\mathbf{R}_i$ are drawn i.i.d. from a zero-mean distribution with finite first four moments. We are interested in estimating the sample covariance matrix $\mathbf{C}_n$ from low-dimensional random projections $\{\mathbf{R}_i^T \mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^m$.

2.7.2 The proposed unbiased covariance estimator

In this section, we present an unbiased estimator for the sample covariance matrix of the full data $\mathbf{C}_n = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T$ from the compressive measurements $\{\mathbf{R}_i^T \mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^m$. Recall that the entries of $\mathbf{R}_i \in \mathbb{R}^{p \times m}$, $m < p$, are drawn i.i.d. from a zero-mean distribution with finite second and fourth moments μ_2 and μ_4, and kurtosis κ. One advantage of our approach compared to the prior work is that our work makes much weaker assumptions on the data. Therefore, our analysis can be used to develop a better understanding of extracting the covariance structure in the compressive domain.

To begin, let's consider a rescaled version of the sample covariance matrix of the projected data $\{\mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^p$

$$\widehat{\mathbf{C}}_n := \frac{1}{(m^2 + m)\mu_2^2} \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{R}_i \mathbf{R}_i^T, \tag{2.21}$$

and we compute the expectation of $\widehat{\mathbf{C}}_n$ based on the following theorem.

Theorem 2.6. *Consider a rescaled version of the sample covariance matrix in the compressive domain*

$$\widehat{\mathbf{C}}_n = \frac{1}{(m^2+m)\mu_2^2} \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{R}_i \mathbf{R}_i^T \tag{2.22}$$

where $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^p$ *are data samples and the entries of* $\{\mathbf{R}_i\}_{i=1}^n \in \mathbb{R}^{p\times m}$, $m < p$, *are drawn i.i.d. from a zero-mean distribution with finite second and fourth moments* μ_2 *and* μ_4, *and kurtosis* κ. *Then, the expectation of* $\widehat{\mathbf{C}}_n$ *over the randomness in* $\{\mathbf{R}_i\}_{i=1}^n$

$$\mathbb{E}[\widehat{\mathbf{C}}_n] = \mathbf{C}_n + \frac{\kappa}{m+1}\mathrm{diag}\left(\mathbf{C}_n\right) + \frac{1}{m+1}\mathrm{tr}\left(\mathbf{C}_n\right)\mathbf{I}_{p\times p} \tag{2.23}$$

where $\mathbf{C}_n = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T$ *and* $\mathrm{diag}(\mathbf{C}_n)$ *denotes the matrix formed by zeroing all but the diagonal entries of* $\mathbf{C}_n$.

Proof. See Appendix A.7. □

We observe that the expectation of the covariance estimator $\widehat{\mathbf{C}}_n$ has three components: the sample covariance matrix of the full data $\mathbf{C}_n$ that we wish to extract and two additional bias terms. The bias term $\frac{\kappa}{m+1}\mathrm{diag}(\mathbf{C}_n)$ can be viewed as representing a bias of the estimator towards the nearest canonical basis vectors. The value of this term depends on both the kurtosis of the distribution used in generating random matrices $\{\mathbf{R}_i\}_{i=1}^n$ and the structure of the underlying covariance matrix $\mathbf{C}_n$. In contrast, the bias term $\frac{1}{m+1}\mathrm{tr}(\mathbf{C}_n)\mathbf{I}_{p\times p}$ is independent of the shape of the distribution. This term shows that the energy of the compressed data is somewhat scattered into different directions in $\mathbb{R}^p$, as $\mathrm{tr}(\mathbf{C}_n)$ represents the energy of the input data. Note that these two bias terms are decreasing functions of m, where m is the number of linear measurements.

The challenge here is to remove these two bias terms by modifying the sample covariance matrix $\widehat{\mathbf{C}}_n$ in the compressive domain. To do this, we first find the expectation of $\mathrm{diag}(\widehat{\mathbf{C}}_n)$ by using linearity of expectation

$$\begin{aligned}\mathbb{E}[\mathrm{diag}(\widehat{\mathbf{C}}_n)] &= \mathrm{diag}(\mathbb{E}[\widehat{\mathbf{C}}_n]) \\ &= \left(1+\frac{\kappa}{m+1}\right)\mathrm{diag}(\mathbf{C}_n) + \frac{\mathrm{tr}\left(\mathbf{C}_n\right)}{m+1}\mathbf{I}_{p\times p}.\end{aligned} \tag{2.24}$$

We also need to compute the expectation of $\mathrm{tr}(\widehat{\mathbf{C}}_n)$

$$\mathbb{E}[\mathrm{tr}(\widehat{\mathbf{C}}_n)] = \mathrm{tr}(\mathbb{E}[\widehat{\mathbf{C}}_n]) = \frac{(m+1+\kappa+p)}{m+1}\,\mathrm{tr}(\mathbf{C}_n) \tag{2.25}$$

since trace is a linear operator. Hence, we can modify $\widehat{\mathbf{C}}_n$ to obtain an unbiased estimator for the sample covariance matrix of the full data

$$\widehat{\mathbf{\Sigma}}_n := \widehat{\mathbf{C}}_n - \alpha_1\,\mathrm{diag}\left(\widehat{\mathbf{C}}_n\right) - \alpha_2\,\mathrm{tr}\left(\widehat{\mathbf{C}}_n\right)\mathbf{I}_{p\times p} \tag{2.26}$$

where

$$\alpha_1 := \frac{\frac{\kappa}{m+1}}{1+\frac{\kappa}{m+1}} \tag{2.27}$$

and

$$\alpha_2 := \frac{1}{(1+\frac{\kappa}{m+1})(m+1+\kappa+p)}. \tag{2.28}$$

We immediately see that $\widehat{\mathbf{\Sigma}}_n$ is an unbiased estimator for the sample covariance matrix of the full data

$$\mathbb{E}[\widehat{\mathbf{\Sigma}}_n] = \mathbf{C}_n. \tag{2.29}$$

Note that α_1 and α_2 are two *known* constants that are used to modify the biased estimator $\widehat{\mathbf{C}}_n$ based on $\mathrm{diag}(\widehat{\mathbf{C}}_n)$ and $\mathrm{tr}(\widehat{\mathbf{C}}_n)$.

2.7.3 Experimental results

In this section, we examine the performance of unbiased covariance estimator $\widehat{\mathbf{\Sigma}}_n$ on three real-world data sets. We compare $\widehat{\mathbf{\Sigma}}_n$ with the biased estimator introduced in (2.21). To evaluate the estimate accuracy, we use the normalized covariance estimation error defined as

$$\text{normalized estimation error}(\widehat{\mathbf{\Sigma}}_n) := \frac{\|\widehat{\mathbf{\Sigma}}_n - \mathbf{C}_n\|_2}{\|\mathbf{C}_n\|_2}.$$

Therefore, this criterion is a measure of closeness of the estimated covariance matrix from compressive measurements to the underlying covariance matrix $\mathbf{C}_n$ based on the spectral

norm. Since the estimation of covariance matrix from compressive measurements is stochastic, we re-run each of the following experiments 100 times and report the mean.

In all experiments, the parameter $\frac{m}{p} = 0.4$ is fixed and we analyze the accuracy of the estimated covariance matrix from the sparse random projections for various values of compression factor $\gamma = \frac{m}{s}$. Note that we are interested in choosing the parameter s such that $\gamma < 1$. In fact, a value of γ closer to zero indicates the case where we achieve substantial reduction of the acquisition and computation cost by increasing the parameter s in the distribution of random matrices.

2.7.3.1 MNIST data set

In the first experiment, we examine the MNIST data set of handwritten digits. This data set consists of centered versions of digits that are stored as a 28×28 pixel image. We use data from digit "0" which contains $n = 6903$ data samples that are vectorized with the dimension $p = 28^2 = 784$. Some examples are shown in Fig. 2.6.

Figure 2.6: Examples from digit "0" in the MNIST data set.

The performance of our unbiased estimator $\widehat{\mathbf{\Sigma}}_n$ is compared with the biased estimator in Fig. 2.7. We see that the proposed estimator $\widehat{\mathbf{\Sigma}}_n$ results in more accurate estimates for all values of the compression factor γ. For example, at $\gamma = 0.1$, our estimator $\widehat{\mathbf{\Sigma}}_n$ decreases the estimation error by almost a factor of 2.

To see the tradeoffs between computational savings and accuracy, the running times to form our unbiased covariance etsimator $\widehat{\mathbf{\Sigma}}_n$ and the biased estimator $\widehat{\mathbf{C}}_n$ are reported in Fig. 2.7. The running times are normalized by the time spent to form the sample covariance matrix of the full data $\mathbf{C}_n$. Both unbiased and biased estimators from compressive measurements show a speedup over $\mathbf{C}_n$ by almost a factor of γ^2. Moreover, we observe that $\widehat{\mathbf{\Sigma}}_n$ has

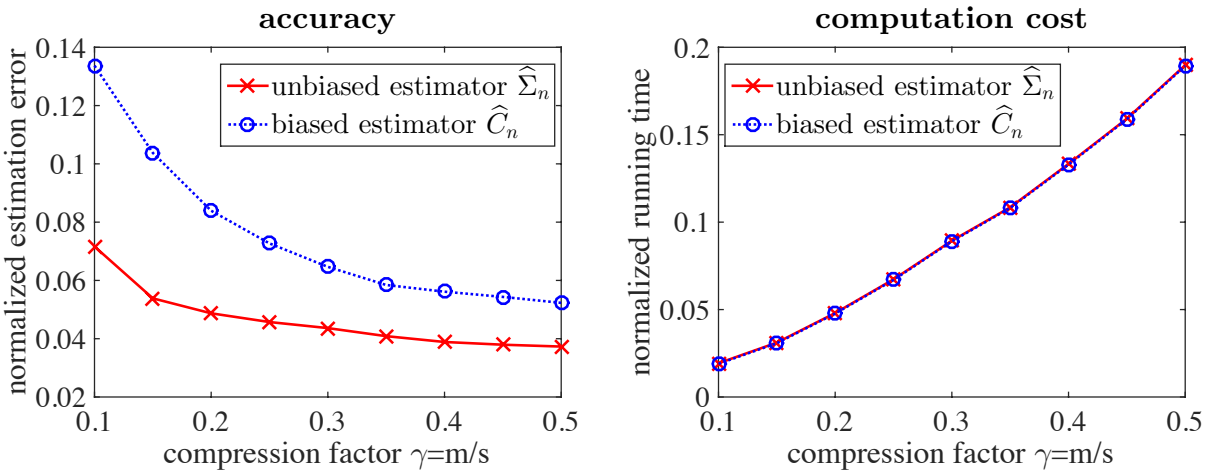


Figure 2.7: Accuracy and computation cost of the estimated covariance matrix from sparse random projections on the MNIST data set. The unbiased estimator $\widehat{\mathbf{\Sigma}}_n$ outperforms the biased estimator for all values of the compression factor γ.

roughly the same running time as $\widehat{\mathbf{C}}_n$. Hence, the additional computation cost to remove the bias terms is negligible. Consequently, our unbiased estimator $\widehat{\mathbf{\Sigma}}_n$ leads to more accurate estimates than the biased estimator $\widehat{\mathbf{C}}_n$ with approximately the same computation cost.

We also show a sample visual comparison of the first eigenvector of the underlying covariance matrix $\mathbf{C}_n$ and that estimated by our approach $\widehat{\mathbf{\Sigma}}_n$ for $\gamma = 0.1$, 0.3, and 0.5. Note that the eigenvectors of covariance matrix encode an orthonormal basis that captures as much of the data's variability as possible. Therefore, we intend to visualize the first eigenvector of $\mathbf{C}_n$ and that estimated by our approach from sparse random projections as a measure of information extraction. As we see in Fig. 2.8, for small values of the compression factor, e.g., $\gamma = 0.1$, the resulting eigenvectors from the full data and compressed data are almost identical. In fact, the first eigenvector of our estimator $\widehat{\mathbf{\Sigma}}_n$ in Fig. 2.8 reveals the underlying structure and pattern in the data set consisting of handwritten digits "0".

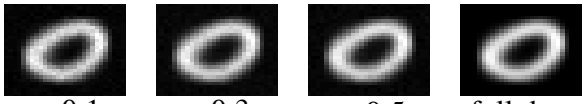


Figure 2.8: Visual comparison between the first eigenvector of the sample covariance matrix $\mathbf{C}_n$ (full data) and that estimated by our proposed estimator $\widehat{\mathbf{\Sigma}}_n$ in the compressive domain for various values of the compression factor γ.

2.7.3.2 Gen4 data set

Our second experiment is with the gen4 data matrix of size 1537×4298 ($p = 1537$ and $n = 4298$) from linear programming problems. This data set is available from the University of Florida Sparse Matrix Collection.

In Fig. 2.9, the accuracy of unbiased estimator $\widehat{\mathbf{\Sigma}}_n$ is compared with the biased estimator. We observe that covariance matrices recovered using our estimator $\widehat{\mathbf{\Sigma}}_n$ are significantly more accurate than those returned by $\widehat{\mathbf{C}}_n$. In fact, our proposed estimator $\widehat{\mathbf{\Sigma}}_n$ decreases the estimation error by an order of magnitude.

Moreover, we examine the tradeoffs between computational savings and accuracy on this data set. The running times to compute $\widehat{\mathbf{\Sigma}}_n$ and $\widehat{\mathbf{C}}_n$ for various values of γ are reported in Fig. 2.9. Recall that the running times are normalized by the time spent to form the sample covariance matrix of the full data $\mathbf{C}_n$. We see that both estimators show a speedup over $\mathbf{C}_n$ by almost a factor of γ^2. Similar to the previous example, the additional computation cost to remove the bias terms is negligible. Therefore, our unbiased estimator $\widehat{\mathbf{\Sigma}}_n$ results in more accurate estimates than $\widehat{\mathbf{C}}_n$ with roughly the same computation cost.

2.7.3.3 Traffic data set

In the last experiment, we consider the traffic data set which contains video surveillance of traffic from a stationary camera. Some examples of this data set are shown in Fig. 2.10. Considering individual frames of video as data samples, each frame is a 48×48 pixel image that is vectorized so $p = 48^2 = 2304$. Moreover, we have access to $n = 5139$ frames.

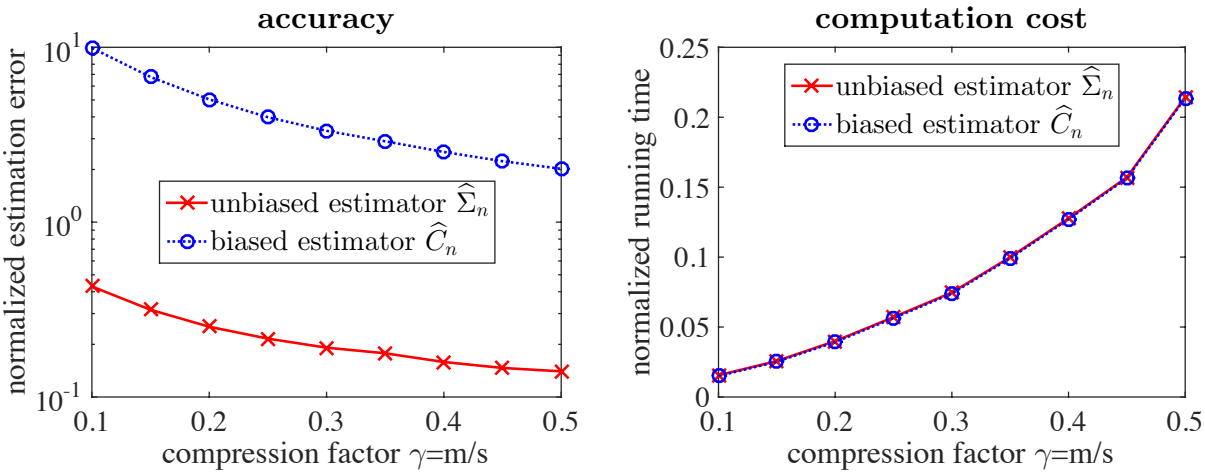


Figure 2.9: Plot of the normalized covariance estimation error (log scale) and computation cost on the gen4 data set. Our estimator $\widehat{\mathbf{\Sigma}}_n$ is compared with the biased estimator for varying values of γ. We see that the unbiased covariance estimator $\widehat{\mathbf{\Sigma}}_n$ decreases the estimation error by an order of magnitude.

Figure 2.10: Example frames of the traffic data set.

Our proposed approach is used to estimate the sample covariance matrix of the full data $\mathbf{C}_n$ from sparse random projections. As we see in Fig. 2.11, using unbiased estimator $\widehat{\mathbf{\Sigma}}_n$ for this data set results in accurate estimates such that the normalized estimation error is less than 0.06 for all values of compression factor γ. The biased estimator $\widehat{\mathbf{C}}_n$ has approximately the same accuracy as $\widehat{\mathbf{\Sigma}}_n$ on this data set. The reason is because the bias term $\frac{\kappa}{m+1}\,\mathrm{diag}(\mathbf{C}_n)$ depends on both the kurtosis κ and the structure of the underlying covariance matrix $\mathbf{C}_n$. Hence, the value of this bias term can be relatively small for some special structures on

$\mathbf{C}_n$. However, our proposed estimator $\widehat{\mathbf{\Sigma}}_n$ provides an unbiased estimate of $\mathbf{C}_n$ without any restrictive assumptions on the data's structure.

Moreover, we examine the tradeoffs between computational savings and accuracy on the traffic data set. We report the running times to compute $\widehat{\mathbf{\Sigma}}_n$ and $\widehat{\mathbf{C}}_n$ normalized by the time spent to form the sample covariance matrix of the full data $\mathbf{C}_n$. As we see in Fig. 2.11, the computational savings are proportional to γ^2. We also observe that $\widehat{\mathbf{\Sigma}}_n$ has roughly the same running time as $\widehat{\mathbf{C}}_n$.

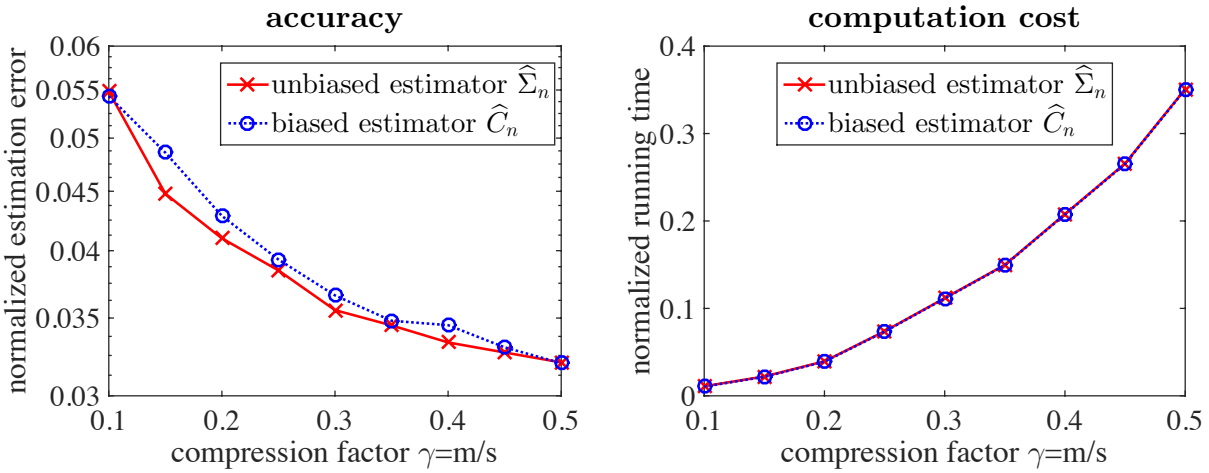


Figure 2.11: Accuracy and computation cost of the estimated covariance matrix from sparse random projections on the traffic data set for varying values of the compression factor γ.

In Fig. 2.12, we provide a sample visual comparison of the first eigenvector of the underlying covariance matrix $\mathbf{C}_n$ and that estimated by our approach $\widehat{\mathbf{\Sigma}}_n$ for $\gamma = 0.1$, 0.3, and 0.5. We see that the first eigenvectors of $\widehat{\mathbf{\Sigma}}_n$ and $\mathbf{C}_n$ are almost identical even for small values of the compression factor, e.g., $\gamma = 0.1$. The first eigenvector of our covariance estimator $\widehat{\mathbf{\Sigma}}_n$ reveals the underlying structure from sparse random projections, which appears to be a traffic trend along the roadway.

This experiment typifies a real-world application of our approach. We showed that our proposed estimator $\widehat{\mathbf{\Sigma}}_n$ can be used to extract the covariance structure of the data from sparse random projections. At the same time, we achieve significant reduction of the computation

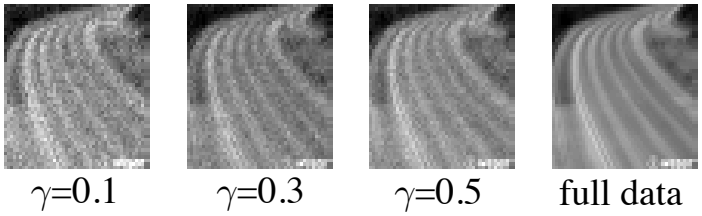


Figure 2.12: Visual comparison between the first eigenvector of the sample covariance matrix $\mathbf{C}_n$ (full data) and that estimated by our approach $\widehat{\mathbf{\Sigma}}_n$ on the traffic data set for the compression factor $\gamma = 0.1$, 0.3, and 0.5.

cost proportional to γ^2, where γ is the compression factor.

As one final note, we observe that our proposed covariance estimator $\widehat{\mathbf{\Sigma}}_n$ results in more accurate estimates on the traffic data set compared to the MNIST and gen4 data sets. To explain this, let us consider the stable rank $\beta := \|\mathbf{C}_n\|_F^2/\|\mathbf{C}_n\|_2^2$, where $\|\mathbf{C}_n\|_F$ represents the Frobenius norm of the sample covariance matrix $\mathbf{C}_n$. The parameter β is a measure that gives us important information about how spread out the eigenvalues of $\mathbf{C}_n$ are. In fact, a value of β closer to 1 indicates a faster rate of decay of the eigenvalues. The values of β for each of the MNIST, gen4, and traffic data sets are 1.03, 1.19, and 1.00 respectively. Therefore, the sample covariance matrix of the traffic data set has one dominant eigenvector. Hence, the experimental results demonstrate that our proposed estimator $\widehat{\mathbf{\Sigma}}_n$ leads to more accurate estimates of $\mathbf{C}_n$ when the eigenvalues of $\mathbf{C}_n$ decay faster, i.e., β is closer to 1.

Chapter 3

Efficient Dictionary Learning via Very Sparse Random Projections

3.1 Introduction

There are several ways to represent low-dimensional structure of high-dimensional data, the best known being principal component analysis (PCA). However, PCA is based on a linear subspace model that is generally not capable of capturing the geometric structure of real-world datasets [20].

The sparse signal model is a nonlinear generalization of the linear subspace model that has been used in various signal and image processing tasks [51, 96, 98], as well as compressive sensing [27]. This model assumes that each data sample can be represented as a linear combination of a few elements (atoms) from a dictionary. Data-adaptive dictionary learning can lead to a much more compact representation than predefined dictionaries such as wavelets, and thus a central problem is finding a good data-adaptive dictionary.

Dictionary learning algorithms such as the method of optimal directions (MOD) [52] and the K-SVD algorithm [6] aim to learn a dictionary by minimizing the representation error of data in an iterative procedure involving two steps of sparse coding and dictionary update. The latter often requires ready access to the entire data available at a central processing unit.

Due to increasing sizes of datasets, not only do algorithms take longer to run, but it may not even be feasible or practical to acquire and/or hold every data entry. In applications such as distributed databases, where data is typically distributed over an interconnected set

of distributed sites [115], it is important to avoid communicating the entire data.

A promising approach to address these issues is to take a compressive sensing approach, where we only have access to compressive measurements of data. In fact, performing signal processing and data mining tasks on compressive versions of the data has been an important topic in the recent literature. For example, in [42], certain inference problems such as detection and estimation within the compressed domain have been studied. Another line of work considers recovery of principal components from compressive measurements [57, 114, 112, 113].

In this chapter, we focus on the problem of dictionary learning based on compressive measurements. Our contributions are twofold. First, we show the connection between dictionary learning in the compressed domain and K-means clustering. Most standard dictionary learning algorithms are indeed a generalization of the K-means clustering algorithm [58], where the reference to K-means is a common approach to analyze the performance of these algorithms [6, 126]. This work takes initial steps towards providing theoretical guarantees for recovery of the true underlying dictionary from compressive measurements. Moreover, our analysis applies to compressive measurements obtained by a general class of random matrices consisting of i.i.d. zero-mean entries and finite first four moments.

Second, we extend the prior work in [110] where compressive dictionary learning for random Gaussian matrices is considered. In particular, we propose a memory and computation efficient dictionary learning algorithm applicable to modern data settings. To do this, we learn a dictionary from very sparse random projections, i.e. projection of the data onto a few very sparse random vectors with Bernoulli-generated nonzero entries. These sparse random projections have been applied in many large-scale applications such as compressive sensing and object tracking [44, 151] and to efficient learning of principal components in the large-scale data setting [113]. To further improve efficiency of our approach, we show how to share the same random matrix across blocks of data samples.

The remainder of this chapter is organized as follows. We first review the related work

on compressive dictionary learning in Section 3.2. Section 3.3 presents connections between compressive dictionary learning and K-means clustering, as well as theoretical guarantees. In Section 3.4, we explain our memory and computation efficient dictionary learning algorithm. Finally, we present experimental results demonstrating the performance and efficiency of our approach in Section 3.5.

3.2 Prior Work on Compressive Dictionary Learning

Several attempts have been made to address the problem of dictionary learning based on compressive measurements. In three roughly contemporary papers [124, 128], and our work [110], three similar algorithms were presented to learn a dictionary based on compressive measurements. Each was inspired by the well-known K-SVD algorithm and closely followed its structure, except in that each aimed to minimize the representation error of the compressive measurements instead of that of the original signals. The exact steps of each algorithm have minor differences, but take a similar overall form.

However, none of these works explicitly aimed at designing the compressive measurements (sketches) to promote the computational efficiency of the resulting compressive K-SVD, so that it would be maximally practical for dictionary learning on large-scale data. Moreover, none of these works gave theoretical performance analysis for such computationally-efficient sketches.

In this work, this is our primary aim, to extend the previous line of work on compressive dictionary learning by developing a memory and computation efficient dictionary learning algorithm. The key to the efficiency of the new scheme is in considering a wider and more general class of random projection matrices for the sketches, including some very sparse ones. We further introduce an initial analysis of the theoretical performance of compressive dictionary learning under these more general random projections.

In this section, we review the general dictionary learning problem and the compressive K-SVD (CK-SVD) algorithm that was introduced in [110] for the case of random Gaussian

matrices. (We note that the approaches of [128] and [124] are similar.) Given a set of n training signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ in $\mathbb{R}^p$, the dictionary learning problem is to find a dictionary $\mathbf{D} \in \mathbb{R}^{p \times K}$ that leads to the best representation under a strict sparsity constraint for each member in the set, i.e., minimizing

$$\min_{\mathbf{D} \in \mathbb{R}^{p \times K}, \mathbf{C} \in \mathbb{R}^{K \times n}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}\mathbf{c}_i\|_2^2 \ s.t. \ \forall i, \ \|\mathbf{c}_i\|_0 \leq T \tag{3.1}$$

where $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_n]$ is the coefficient matrix and the ℓ_0 pseudo-norm $\|\mathbf{c}_i\|_0$ counts the number of nonzero entries of the coefficient vector $\mathbf{c}_i \in \mathbb{R}^K$. Moreover, the columns of the dictionary $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K]$ are typically assumed to have unit ℓ_2-norm. Problem (3.1) is generally intractable so we look for approximate solutions (e.g., via K-SVD [6]).

We then consider compressed measurements (sketches), where each measurement is obtained by taking inner products of the data sample $\mathbf{x}_i \in \mathbb{R}^p$ with the columns of a matrix $\mathbf{R}_i$, i.e., $\mathbf{y}_i = \mathbf{R}_i^T \mathbf{x}_i$ with $\{\mathbf{R}_i\}_{i=1}^n \in \mathbb{R}^{p \times m}$, $m < p$, and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{m \times n}$. In [110], the entries of $\mathbf{R}_i$ are i.i.d. from a zero-mean Gaussian distribution, which is an assumption we drop in the current work.

Given access only to the compressed measurements $\mathbf{y}_i$ and not $\mathbf{x}_i$, we attempt to solve the following compressive dictionary learning problem:

$$\min_{\mathbf{D} \in \mathbb{R}^{p \times K}, \mathbf{C} \in \mathbb{R}^{K \times n}} \sum_{i=1}^{n} \left\|\mathbf{y}_i - \mathbf{R}_i^T \mathbf{D}\mathbf{c}_i\right\|_2^2 \ s.t. \ \forall i, \ \|\mathbf{c}_i\|_0 \leq T. \tag{3.2}$$

In the CK-SVD algorithm, the objective function in (3.2) is minimized in a simple iterative approach that alternates between sparse coding and dictionary update steps.

3.2.1 Sparse coding

In the sparse coding step, the penalty term in (3.2) is minimized with respect to a fixed $\mathbf{D}$ to find the coefficient matrix $\mathbf{C}$ under the strict sparsity constraint. This can be written

as

$$\min_{\mathbf{C}\in\mathbb{R}^{K\times n}} \sum_{i=1}^{n} \|\mathbf{y}_i - \Psi_i \mathbf{c}_i\|_2^2 \ \ s.t.\ \forall i,\ \|\mathbf{c}_i\|_0 \leq T \tag{3.3}$$

where $\Psi_i = \mathbf{R}_i^T \mathbf{D} \in \mathbb{R}^{m\times K}$ is a fixed equivalent dictionary for representation of $\mathbf{y}_i$. This optimization problem can be considered as n distinct optimization problems for each compressive measurement. We can then use a variety of algorithms, such as OMP, to find the approximate solution $\mathbf{c}_i$ [135].

3.2.2 Dictionary update

The approach is to update the k^{th} dictionary atom $\mathbf{d}_k$ and its corresponding coefficients while holding $\mathbf{d}_j$ fixed for $j \neq k$, and then repeat for $k+1$, until $k = K$. The penalty term in (3.2) can be written as

$$\begin{aligned}
&\sum_{i=1}^{n} \left\| \mathbf{y}_i - \mathbf{R}_i^T \sum_{j=1}^{K} c_{i,j}\mathbf{d}_j \right\|_2^2 \\
&= \sum_{i=1}^{n} \left\| \left(\mathbf{y}_i - \mathbf{R}_i^T \sum_{j\neq k} c_{i,j}\mathbf{d}_j \right) - c_{i,k}\mathbf{R}_i^T \mathbf{d}_k \right\|_2^2 \\
&= \sum_{i\in\mathcal{I}_k} \left\| \mathbf{e}_{i,k} - c_{i,k}\mathbf{R}_i^T \mathbf{d}_k \right\|_2^2 + \sum_{i\notin\mathcal{I}_k} \|\mathbf{e}_{i,k}\|_2^2
\end{aligned} \tag{3.4}$$

where $c_{i,k}$ is the k^{th} element of $\mathbf{c}_i \in \mathbb{R}^K$, $\mathcal{I}_k$ is a set of indices of compressive measurements for which $c_{i,k} \neq 0$, and $\mathbf{e}_{i,k} = \mathbf{y}_i - \mathbf{R}_i^T \sum_{j\neq k} c_{i,j}\mathbf{d}_j \in \mathbb{R}^m$ is the representation error for $\mathbf{y}_i$ when the k^{th} dictionary atom is removed. The objective in (3.4) is a quadratic function of $\mathbf{d}_k$ and the minimizer is obtained by setting the derivative with respect to $\mathbf{d}_k$ equal to zero. Hence,

$$\mathbf{G}_k \mathbf{d}_k = \mathbf{b}_k \tag{3.5}$$

where $\mathbf{G}_k = \sum_{i\in\mathcal{I}_k} c_{i,k}^2 \mathbf{R}_i \mathbf{R}_i^T$ and $\mathbf{b}_k = \sum_{i\in\mathcal{I}_k} c_{i,k}\mathbf{R}_i \mathbf{e}_{i,k}$. Therefore, we get the closed-form solution $\mathbf{d}_k = \mathbf{G}_k^+ \mathbf{b}_k$, where $\mathbf{G}_k^+$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{G}_k$. Once

given the new $\mathbf{d}_k$, the optimal $c_{i,k}$ for each $i \in \mathcal{I}_k$ is given by least squares as

$$c_{i,k} = \frac{\langle \mathbf{e}_{i,k}, \mathbf{R}_i^T \mathbf{d}_k \rangle}{\|\mathbf{R}_i^T \mathbf{d}_k\|_2^2}. \tag{3.6}$$

By design, the support of the coefficient matrix $\mathbf{C}$ is preserved, just as in the K-SVD algorithm. Note that we are departing with K-SVD here, since K-SVD updates dictionary atoms and corresponding coefficients simultaneously via singular value decomposition.

3.3 Initial Theoretical Analysis: K-means Case

In this section, we provide an initial theoretical analysis of the performance of the CK-SVD algorithm by restricting our attention to a special case of dictionary learning: K-means clustering. In this special case, we can provide theoretical guarantees on the performance of CK-SVD at every step, in relation to the steps of K-means. Moreover, these guarantees will hold for a very general class of projection matrices including very sparse random projections.

We consider a statistical framework to establish the connection between CK-SVD and K-means. K-means clustering can be viewed as a special case of dictionary learning in which each data sample is allowed to use one dictionary atom (cluster center), i.e. $T = 1$, and the corresponding coefficient is set to be 1. Therefore, we consider the following generative model

$$\mathbf{x}_i = \overline{\mathbf{d}_k} + \epsilon_i, \ i \in \mathcal{I}_k \tag{3.7}$$

where $\overline{\mathbf{d}_k}$ is the center of the k^{th} cluster, and $\{\epsilon_i\}_{i=1}^n \in \mathbb{R}^p$ represent residuals in signal approximation and they are drawn i.i.d. from $\mathcal{N}(\mathbf{0}, \frac{\sigma^2}{p}\mathbf{I}_{p\times p})$, so that the approximation error is $\mathbb{E}[\|\epsilon\|_2^2] = \sigma^2$. The set $\{\mathcal{I}_k\}_{k=1}^K$ is an arbitrary partition of $(1, 2, \ldots, n)$, with the condition that $|\mathcal{I}_k| \to \infty$ as $n \to \infty$. The random matrices $\mathbf{R}_i$ are assumed to satisfy the following:

Assumption 1. *Each entry of the random matrices $\{\mathbf{R}_i\}_{i=1}^n \in \mathbb{R}^{p\times m}$ is drawn i.i.d. from a general class of zero-mean distributions with finite first four moments $\{\mu_k\}_{k=1}^4$.*

We will see that the distribution's kurtosis is a key factor in our results. The kurtosis, defined as $\kappa \triangleq \frac{\mu_4}{\mu_2^2} - 3$, is a measure of peakedness and heaviness of tail for a distribution.

We now show how, in this special case of K-means, CK-SVD would update the cluster centers. As mentioned before, in this case, we should set $T = 1$ and the corresponding coefficients are set to be 1. This means that for all $i \in \mathcal{I}_k$, we have $c_{i,k} = 1$, and $c_{i,j} = 0$ for $j \neq k$, and it leads to $\mathbf{e}_{i,k} = \mathbf{y}_i, \forall i \in \mathcal{I}_k$. Then, the update formula for the k^{th} dictionary atom of CK-SVD given in (3.5) reduces to

$$\left(\sum_{i\in\mathcal{I}_k} \mathbf{R}_i\mathbf{R}_i^T\right)\mathbf{d}_k = \sum_{i\in\mathcal{I}_k} \mathbf{R}_i\mathbf{y}_i. \tag{3.8}$$

Hence, similar to K-means, the process of updating K dictionary atoms becomes independent of each other. We can rewrite (3.8) as $\mathbf{H}_k\mathbf{d}_k = \mathbf{f}_k$, where

$$\mathbf{H}_k \triangleq \frac{1}{m\mu_2}\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k} \mathbf{R}_i\mathbf{R}_i^T,\ \mathbf{f}_k \triangleq \frac{1}{m\mu_2}\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k} \mathbf{R}_i\mathbf{y}_i. \tag{3.9}$$

In Theorem 2.1, it is shown that $\mathbb{E}[\mathbf{R}_i\mathbf{R}_i^T] = m\mu_2\mathbf{I}_{p\times p}$. Thus, we see

$$\begin{aligned}\mathbb{E}\left[\mathbf{R}_i\mathbf{y}_i\right] &= \mathbb{E}\left[\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\right] \\ &= \mathbb{E}\left[\mathbf{R}_i\mathbf{R}_i^T\overline{\mathbf{d}_k}\right] + \mathbb{E}\left[\mathbf{R}_i\mathbf{R}_i^T\epsilon_i\right] \\ &= \mathbb{E}\left[\mathbf{R}_i\mathbf{R}_i^T\right]\overline{\mathbf{d}_k} + \mathbb{E}\left[\mathbf{R}_i\mathbf{R}_i^T\right]\mathbb{E}\left[\epsilon_i\right] = m\mu_2\overline{\mathbf{d}_k}.\end{aligned} \tag{3.10}$$

Therefore, when the number of samples is sufficiently large, using the law of large numbers, $\mathbf{H}_k$ and $\mathbf{f}_k$ converge to $\frac{1}{m\mu_2}\mathbb{E}[\mathbf{R}_i\mathbf{R}_i^T] = \mathbf{I}_{p\times p}$ and $\frac{1}{m\mu_2}\mathbb{E}[\mathbf{R}_i\mathbf{y}_i] = \overline{\mathbf{d}_k}$. Hence, the updated dictionary atom in our CK-SVD is the original center of cluster, i.e. $\mathbf{d}_k = \overline{\mathbf{d}_k}$, exactly as in K-means. Note that in this case even one measurement per signal $m = 1$ is sufficient.

The following theorem characterizes convergence rates for $\mathbf{H}_k$ and $\mathbf{f}_k$ based on various parameters such as the number of samples and the choice of random matrices.

Theorem 3.1. *Assume Assumption 1. Then, $\mathbf{H}_k$ defined in (3.9) converges to the identity matrix $\mathbf{I}_{p\times p}$ and for any $\eta > 0$, we have*

$$\mathbb{P}\left(\frac{\|\mathbf{H}_k - \mathbf{I}_{p\times p}\|_F}{\|\mathbf{I}_{p\times p}\|_F} \leq \eta\right) \geq 1 - P_0 \tag{3.11}$$

where

$$P_0 = \frac{1}{m\left|\mathcal{I}_k\right|\eta^2}\left(\kappa + 1 + p\right). \tag{3.12}$$

Also, consider the probabilistic model given in (3.7) and compressive measurements $\mathbf{y}_i = \mathbf{R}_i^T\mathbf{x}_i$*. Then,* $\mathbf{f}_k$ *defined in (3.9) converges to the center of original data and for any* $\eta > 0$*, we have*

$$\mathbb{P}\left(\frac{\left\|\mathbf{f}_k - \overline{\mathbf{d}_k}\right\|_2}{\left\|\overline{\mathbf{d}_k}\right\|_2} \leq \eta\right) \geq 1 - P_1 \tag{3.13}$$

where

$$P_1 = P_0 + \frac{1}{SNR}\left(P_0 + \frac{1}{\left|\mathcal{I}_k\right|\eta^2}\right) \tag{3.14}$$

and the signal-to-noise ratio is defined as $SNR \triangleq \frac{\left\|\overline{\mathbf{d}_k}\right\|_2^2}{\sigma^2}$.

Proof. The matrix $\mathbf{H}_k$ can be written as $\mathbf{H}_k = \frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\mathbf{Q}_i$, where $\mathbf{Q}_i \triangleq \frac{1}{m\mu_2}\mathbf{R}_i\mathbf{R}_i^T$. We have $\mathbb{E}[\mathbf{Q}_i] = \mathbf{I}_{p\times p}$ and hence $\mathbb{E}[\mathbf{H}_k] = \mathbf{I}_{p\times p}$. We find the deviation of $\mathbf{H}_k$ from its mean value:

$$\begin{aligned}
&\mathbb{E}\left[\left\|\mathbf{H}_k - \mathbf{I}_{p\times p}\right\|_F^2\right] = \mathbb{E}\left[\operatorname{tr}\left(\left(\mathbf{H}_k - \mathbf{I}_{p\times p}\right)\left(\mathbf{H}_k - \mathbf{I}_{p\times p}\right)\right)\right] \\
&= \frac{1}{\left|\mathcal{I}_k\right|^2}\operatorname{tr}\left(\mathbb{E}\left[\sum_{i\in\mathcal{I}_k}\left(\mathbf{Q}_i - \mathbf{I}_{p\times p}\right)\sum_{i\in\mathcal{I}_k}\left(\mathbf{Q}_i - \mathbf{I}_{p\times p}\right)\right]\right) \\
&= \frac{1}{\left|\mathcal{I}_k\right|^2}\operatorname{tr}\left(\mathbb{E}\left[\sum_{i\in\mathcal{I}_k}\left(\mathbf{Q}_i - \mathbf{I}_{p\times p}\right)\left(\mathbf{Q}_i - \mathbf{I}_{p\times p}\right)\right]\right) \\
&= \frac{1}{\left|\mathcal{I}_k\right|}\left(\operatorname{tr}\left(\mathbb{E}\left[\mathbf{Q}^2\right]\right) - p\right).
\end{aligned} \tag{3.15}$$

The term $\operatorname{tr}(\mathbb{E}[\mathbf{Q}^2])$ can be computed as

$$\begin{aligned}
\operatorname{tr}\left(\mathbb{E}\left[\mathbf{Q}^2\right]\right) &= \frac{1}{m^2\mu_2^2}\operatorname{tr}\left(\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\mathbf{R}\mathbf{R}^T\right]\right) \\
&\overset{(a)}{=} \left(\frac{\kappa+1}{m} + \frac{p}{m} + 1\right)p
\end{aligned} \tag{3.16}$$

where (a) follows from

$$\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\mathbf{R}\mathbf{R}^T\right] = m\mu_2^2\left(\kappa + m + p + 1\right)\mathbf{I}_{p\times p}. \tag{3.17}$$

Hence, we get

$$\mathbb{E}\left[\left\|\mathbf{H}_k - \mathbf{I}_{p\times p}\right\|_F^2\right] = \frac{1}{\left|\mathcal{I}_k\right|}\left(\frac{\kappa+1}{m} + \frac{p}{m}\right)p. \tag{3.18}$$

Note that the Frobenius norm of a matrix is equivalent to the ℓ_2-norm of its vector form. Therefore, we can now use the Chebyshev inequality

$$\begin{aligned}\mathbb{P}\left(\left\|\mathbf{H}_k - \mathbf{I}_{p\times p}\right\|_F \geq \eta \left\|\mathbf{I}_{p\times p}\right\|_F\right) &\leq \frac{\mathbb{E}\left[\left\|\mathbf{H}_k - \mathbf{I}_{p\times p}\right\|_F^2\right]}{\eta^2 \left\|\mathbf{I}_{p\times p}\right\|_F^2} \\ &= \frac{1}{m\left|\mathcal{I}_k\right|\eta^2}\left(\kappa + 1 + p\right). \end{aligned} \tag{3.19}$$

Next, we find the variance of $\mathbf{f}_k = \frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{y}_i$. Because $\mathbb{E}[\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{y}_i] = \overline{\mathbf{d}_k}$, we see that $\mathbf{f}_k$ is an unbiased estimator for the center, i.e. $\mathbb{E}[\mathbf{f}_k] = \overline{\mathbf{d}_k}$. Now, we have

$$\begin{aligned}&\mathbb{E}\left[\left\|\mathbf{f}_k - \overline{\mathbf{d}_k}\right\|_2^2\right] = \operatorname{tr}\left(\mathbb{E}\left[\left(\mathbf{f}_k - \overline{\mathbf{d}_k}\right)\left(\mathbf{f}_k - \overline{\mathbf{d}_k}\right)^T\right]\right) \\ &= \frac{1}{\left|\mathcal{I}_k\right|^2}\operatorname{tr}\left(\mathbb{E}\left[\sum_{i\in\mathcal{I}_k}\left(\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{y}_i - \overline{\mathbf{d}_k}\right)\left(\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{y}_i - \overline{\mathbf{d}_k}\right)^T\right]\right) \\ &= \frac{1}{\left|\mathcal{I}_k\right|}\operatorname{tr}\left(\mathbb{E}\left[\left(\frac{1}{m\mu_2}\mathbf{R}\mathbf{y} - \overline{\mathbf{d}_k}\right)\left(\frac{1}{m\mu_2}\mathbf{R}\mathbf{y} - \overline{\mathbf{d}_k}\right)^T\right]\right) \\ &= \frac{1}{\left|\mathcal{I}_k\right|}\left(\frac{1}{m^2\mu_2^2}\operatorname{tr}\left(\mathbb{E}\left[\mathbf{R}\mathbf{y}\mathbf{y}^T\mathbf{R}^T\right]\right) - \left\|\overline{\mathbf{d}_k}\right\|_2^2\right). \end{aligned} \tag{3.20}$$

We can compute $\operatorname{tr}\left(\mathbb{E}\left[\mathbf{R}\mathbf{y}\mathbf{y}^T\mathbf{R}^T\right]\right)$ as follows

$$\begin{aligned}&\operatorname{tr}\left(\mathbb{E}\left[\mathbf{R}\mathbf{y}\mathbf{y}^T\mathbf{R}^T\right]\right) = \operatorname{tr}\left(\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T\right]\right) \\ &= \operatorname{tr}\left(\mathbb{E}_{\mathbf{R}}\left[\mathbf{R}\mathbf{R}^T\mathbb{E}_{\mathbf{x}}\left[\mathbf{x}\mathbf{x}^T\right]\mathbf{R}\mathbf{R}^T\right]\right) \\ &= \operatorname{tr}\left(\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\overline{\mathbf{d}_k\mathbf{d}_k}^T\mathbf{R}\mathbf{R}^T\right]\right) + \frac{\sigma^2}{p}\operatorname{tr}\left(\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\mathbf{R}\mathbf{R}^T\right]\right) \\ &\overset{(b)}{=} m^2\mu_2^2\left(\frac{\kappa}{m} + \frac{p}{m} + \frac{1}{m} + 1\right)\left(\left\|\overline{\mathbf{d}_k}\right\|_2^2 + \sigma^2\right) \end{aligned} \tag{3.21}$$

where (b) follows from (3.17) and the following result for a fixed vector $\mathbf{v} \in \mathbb{R}^p$:

$$\operatorname{tr}\left(\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\mathbf{v}\mathbf{v}^T\mathbf{R}\mathbf{R}^T\right]\right) = \mu_2^2\left\|\mathbf{v}\right\|_2^2\left(\left(m^2 + m\right) + mp + \kappa m\right).$$

Hence, we substitute (3.21) into (3.20) to find the variance

$$\begin{aligned}&\mathbb{E}\left[\left\|\mathbf{f}_k - \overline{\mathbf{d}_k}\right\|_2^2\right] \\ &= \frac{\left\|\overline{\mathbf{d}_k}\right\|_2^2}{\left|\mathcal{I}_k\right|}\left(\left(\frac{\kappa+1}{m} + \frac{p}{m}\right) + \frac{\sigma^2}{\left\|\overline{\mathbf{d}_k}\right\|_2^2}\left(\frac{\kappa+1}{m} + \frac{p}{m} + 1\right)\right). \end{aligned}$$

Using the variance in the Chebyshev inequality yields the given convergence rate. This completes the proof. □

We see that for a fixed error bound η, as $|\mathcal{I}_k|$ increases, the error probability P_0 decreases at rate $\frac{1}{|\mathcal{I}_k|}$. Therefore, for any fixed $\eta > 0$, the error probability P_0 goes to zero as $|\mathcal{I}_k| \to \infty$. Note that the shape of distribution, specified by the kurtosis, is an important factor. For random matrices with heavy-tailed entries, the error probability P_0 increases. However, P_0 gives us an explicit tradeoff between $|\mathcal{I}_k|$, the measurement ratio, and anisotropy in the distribution. For example, the increase in kurtosis can be compensated by increasing $|\mathcal{I}_k|$. The convergence rate analysis for $\mathbf{f}_k$ follows the same path. We further note that P_1 is a decreasing function of the signal-to-noise ratio and as SNR increases, P_1 gets closer to P_0, where for the case that SNR $\to \infty$, then $P_1 \approx P_0$.

Let's consider an example to gain intuition on the choice of random matrices. We are interested in comparing the dense random Gaussian matrices with very sparse random matrices, where each entry is drawn from $\{-1, 0, +1\}$ with probabilities $\{\frac{1}{2s}, 1 - \frac{1}{s}, \frac{1}{2s}\}$ for $s \geq 1$ (we refer to this distribution as a sparse-Bernoulli distribution with parameter s). $\{\mathbf{R}_i\}_{i=1}^{n} \in \mathbb{R}^{p \times m}$, $p = 100$ and $m/p = 0.3$, are generated with i.i.d. entries both for Gaussian and the sparse-Bernoulli distribution. In Fig. 3.1, we see that as $|\mathcal{I}_k|$ increases, $\mathbf{H}_k$ gets closer to the identity matrix $\mathbf{I}_{p \times p}$. Also, for a fixed $|\mathcal{I}_k|$, as the sparsity of random matrices increases, the kurtosis $\kappa = s - 3$ increases. Therefore, based on Theorem 3.1, we expect that the distance between $\mathbf{H}_k$ and $\mathbf{I}_{p \times p}$ increases. Note that for Gaussian and the sparse-Bernoulli with $s = 3$, we have $\kappa = 0$.

As a final note, our theoretical analysis gives us valuable insight about the number of distinct random matrices required. Based on Theorem 3.1, there is an inherent tradeoff between the accuracy and the number of distinct random matrices used. For example, if we only use one random matrix, we are not able to recover the true dictionary as observed in [63]. Also, increasing the number of distinct random matrices improves the accuracy,

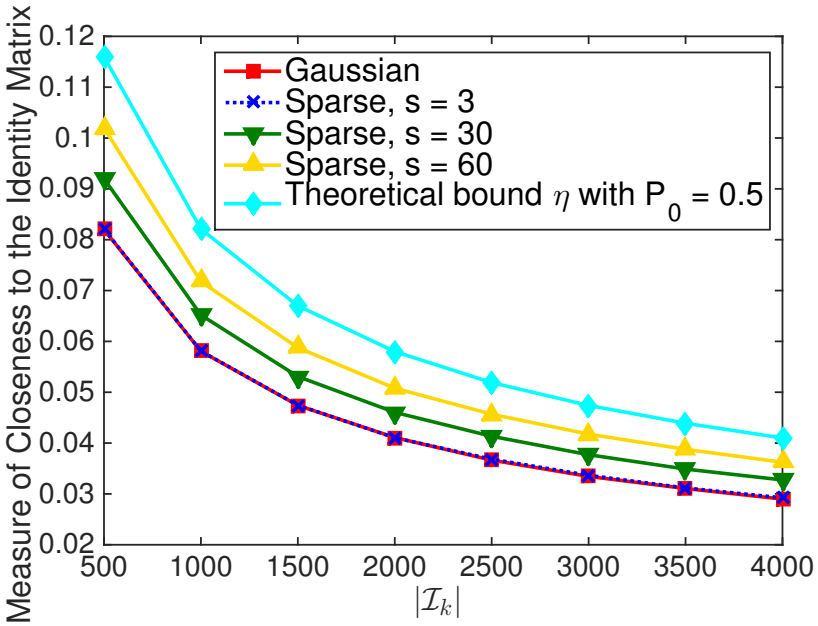


Figure 3.1: Closeness of $\mathbf{H}_k$ to $\mathbf{I}_{p\times p}$ defined as $\|\mathbf{H}_k - \mathbf{I}_{p\times p}\|_F / \|\mathbf{I}_{p\times p}\|_F$. $\{\mathbf{R}_i\}_{i=1}^n \in \mathbb{R}^{p\times m}$, $p = 100$ and $m/p = 0.3$, are generated with i.i.d. entries both for Gaussian and the sparse-Bernoulli distribution. We see that as $|\mathcal{I}_k|$ increases, $\mathbf{H}_k$ gets closer to $\mathbf{I}_{p\times p}$. Also, for fixed $|\mathcal{I}_k|$, as the sparsity of random matrices increases, the kurtosis $\kappa = s - 3$ increases and consequently the distance between $\mathbf{H}_k$ and $\mathbf{I}_{p\times p}$ increases. For Gaussian and the sparse-Bernoulli with $s = 3$, we have $\kappa = 0$. We also plot the theoretical bound η with $P_0 = 0.5$ for the Gaussian case.

as mentioned in [124]. Hence, we can reduce the number of distinct random matrices in large-scale problems where $n = \mathcal{O}(p)$ with controlled loss in accuracy.

3.4 Memory and Computation Efficient Dictionary Learning

Now, we return our attention to general dictionary learning. Inspired by the generality of the projection matrices in Thm. 3.1, we sketch using very sparse random matrices, and furthermore reduce the number of distinct random matrices to increase the efficiency of our approach.

Assume that the original data samples are divided into L blocks $\mathbf{X} = [\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(L)}]$, where $\mathbf{X}^{(l)}$ represents the l^{th} block. Let $\mathbf{R}_l \in \mathbb{R}^{p\times m}$, $m < p$, represent the random matrix

used for the l^{th} block. Then, we have

$$\mathbf{Y}^{(l)} = \mathbf{R}_l^T \mathbf{X}^{(l)},\ 1 \leq l \leq L \tag{3.22}$$

where $\mathbf{Y}^{(l)}$ is the sketch of $\mathbf{X}^{(l)}$. Each entry of $\{\mathbf{R}_l\}_{l=1}^L$ is distributed on $\{-1, 0, +1\}$ with probabilities $\{\frac{1}{2s}, 1-\frac{1}{s}, \frac{1}{2s}\}$. Here, the parameter s controls the sparsity of random matrices such that each column of $\{\mathbf{R}_l\}_{l=1}^L$ has $\frac{p}{s}$ nonzero entries, on average. We are specifically interested in choosing m and s such that the *compression factor* $\gamma \triangleq \frac{m}{s} < 1$. Thus, the cost to acquire each compressive measurement is $\mathcal{O}(\gamma p)$, $\gamma < 1$, vs. the cost for collecting every data entry $O(p)$.

Similarly, we aim to minimize the representation error as

$$\min_{\mathbf{D}\in\mathbb{R}^{p\times K}, \mathbf{C}\in\mathbb{R}^{K\times n}} \sum_{l=1}^{L} \left\|\mathbf{Y}^{(l)} - \mathbf{R}_l^T \mathbf{D}\mathbf{C}^{(l)}\right\|_F^2 \ s.t.\ \forall i,\ \left\|\mathbf{c}_i^{(l)}\right\|_0 \leq T \tag{3.23}$$

where $\mathbf{c}_i^{(l)}$ represents the i^{th} sample in the l^{th} block of the coefficient matrix $\mathbf{C}^{(l)}$. As before, the penalty term in (3.23) is minimized in a simple iterative approach involving two steps. The first step, sparse coding, is the same as the CK-SVD algorithm previously described, except we can take efficient of the block structure and use Batch-OMP [118] in each block which is significantly faster than OMP for each $\mathbf{c_i}$ separately.

3.4.1 Dictionary update

The goal is to update the k^{th} dictionary atom $\mathbf{d}_k$ for $k = 1, \ldots, K$, while assuming that $\mathbf{d}_j$, $j \neq k$, is fixed. The penalty term in (3.23) can be written as

$$\begin{aligned}
\sum_{l=1}^{L} \left\|\mathbf{Y}^{(l)} - \mathbf{R}_l^T \mathbf{D}\mathbf{C}^{(l)}\right\|_F^2 &= \sum_{l=1}^{L}\sum_{i=1}^{n_l} \left\|\mathbf{y}_i^{(l)} - \mathbf{R}_l^T \sum_{j=1}^{K} c_{i,j}^{(l)} \mathbf{d}_j\right\|_2^2 \\
&= \sum_{l=1}^{L}\sum_{i=1}^{n_l} \left\|\left(\mathbf{y}_i^{(l)} - \mathbf{R}_l^T \sum_{j\neq k} c_{i,j}^{(l)} \mathbf{d}_j\right) - c_{i,k}^{(l)} \mathbf{R}_l^T \mathbf{d}_k\right\|_2^2 \\
&= \sum_{l=1}^{L}\sum_{i=1}^{n_l} \left\|\mathbf{e}_{i,k}^{(l)} - c_{i,k}^{(l)} \mathbf{R}_l^T \mathbf{d}_k\right\|_2^2
\end{aligned} \tag{3.24}$$

where $c_{i,k}^{(l)}$ is the k^{th} element of $\mathbf{c}_i^{(l)} \in \mathbb{R}^K$, and $\mathbf{e}_{i,k}^{(l)} \in \mathbb{R}^m$ is the representation error for the compressive measurement $\mathbf{y}_i^{(l)}$ when the k^{th} dictionary atom is removed. The objective function in (3.24) is a quadratic function of $\mathbf{d}_k$ and the minimizer is obtained by setting the derivative of the objective function with respect to $\mathbf{d}_k$ equal to zero. First, let us define $\mathcal{I}_k^{(l)}$ as a set of indices of compressive measurements in the l^{th} block using $\mathbf{d}_k$. Therefore, we get the following expression

$$\mathbf{G}_k \mathbf{d}_k = \mathbf{b}_k \tag{3.25}$$

where

$$\mathbf{G}_k \triangleq \sum_{l=1}^{L} s_k^{(l)} \mathbf{R}_l \mathbf{R}_l^T, \ \mathbf{b}_k \triangleq \sum_{l=1}^{L} \sum_{i \in \mathcal{I}_k^{(l)}} c_{i,k}^{(l)} \mathbf{R}_l \mathbf{e}_{i,k}^{(l)} \tag{3.26}$$

and $s_k^{(l)}$ is defined as the sum of squares of all the coefficients related to the k^{th} dictionary atom in the l^{th} block

$$s_k^{(l)} \triangleq \sum_{i \in \mathcal{I}_k^{(l)}} (c_{i,k}^{(l)})^2. \tag{3.27}$$

Note that $\mathbf{G}_k$ can be computed efficiently: concatenate $\{\mathbf{R}_l\}_{l=1}^L$ in a matrix $\mathbf{R} \triangleq [\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_L] \in \mathbb{R}^{p \times (mL)}$, and define the diagonal matrix $\mathbf{S}_k$ as

$$\mathbf{S}_k \triangleq \operatorname{diag}\Big(\Big[\underbrace{s_k^{(1)}, \ldots, s_k^{(1)}}_{\text{repeated } m \text{ times}}, \ldots, \underbrace{s_k^{(L)}, \ldots, s_k^{(L)}}_{\text{repeated } m \text{ times}} \Big] \Big) \tag{3.28}$$

where $\operatorname{diag}(\mathbf{z})$ represents a square diagonal matrix with the elements of vector $\mathbf{z}$ on the main diagonal. Then, we have $\mathbf{G}_k = \mathbf{R}\mathbf{S}_k\mathbf{R}^T$.

Given the updated $\mathbf{d}_k$, the optimal $c_{i,k}^{(l)}$, for all $i \in \mathcal{I}_k^{(l)}$, is given by least squares as

$$c_{i,k}^{(l)} = \frac{\langle \mathbf{e}_{i,k}^{(l)}, \mathbf{R}_l^T \mathbf{d}_k \rangle}{\|\mathbf{R}_l^T \mathbf{d}_k\|_2^2}, \ \forall i \in \mathcal{I}_k^{(l)}. \tag{3.29}$$

3.5 Experimental Results

We examine the performance of our dictionary learning algorithm on a synthetic dataset. Our proposed method is compared with the fast and efficient implementation of

K-SVD known as Approximate K-SVD (AK-SVD) [118] that requires access to the entire data. We generate $K = 15$ dictionary atoms in $\mathbb{R}^p$, $p = 1000$, drawn from the uniform distribution and normalized to have unit norm. A set of data samples $\{\mathbf{x}_i\}_{i=1}^{50,000} \in \mathbb{R}^p$ is generated where each sample is a linear combination of three distinct atoms, i.e. $T = 3$, and the corresponding coefficients are chosen i.i.d. from the Gaussian distribution $\mathcal{N}(0, 100)$. Then, each data is corrupted by Gaussian noise drawn from $\mathcal{N}(0, 0.04\mathbf{I}_{p\times p})$.

CK-SVD is applied on the set of compressive measurements obtained by very sparse random matrices for various values of the compression factor $\gamma = \frac{1}{3}, \frac{1}{5}, \frac{1}{10}, \frac{1}{20}, \frac{1}{30}$. We set the number of blocks $L = 250$ and $m/p = 0.1$. Performance is evaluated by the magnitude of the inner product between learned and true atoms. A value greater than 0.95 is counted as a successful recovery. Fig. 3.2 shows the results of CK-SVD averaged over 50 independent trials. In practice, when T is small, the updates for $\mathbf{d}_k$ are nearly decoupled, and we may delay updating $c_{i,k}^{(l)}$ until after all K updates of $\mathbf{d}_k$. For $T = 3$, the accuracy results are indistinguishable.

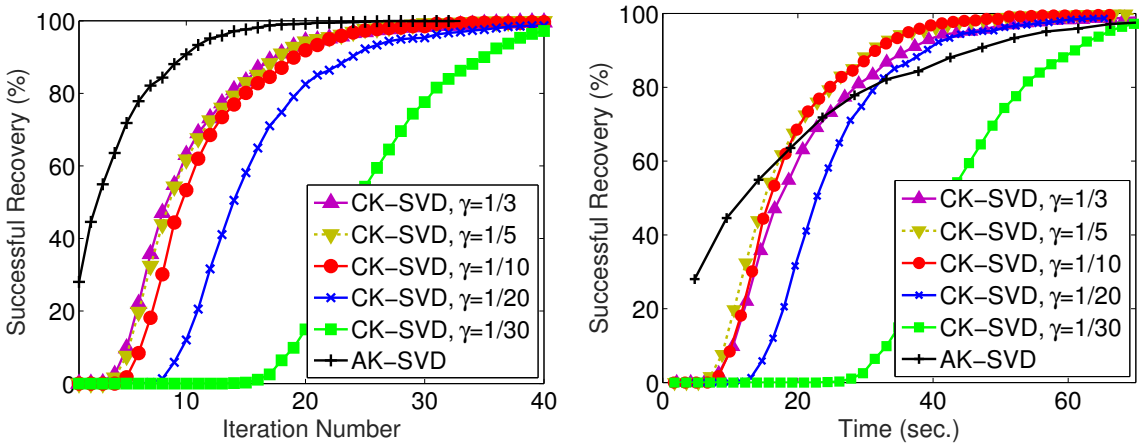


Figure 3.2: Results for synthetic data. Plot of successful recovery vs. (a) iteration number, and (b) time. Our method CK-SVD for varying compression factor γ is compared with AK-SVD. We observe that our method is both memory/computation efficient and accurate for $\gamma = \frac{1}{5}$ and $\gamma = \frac{1}{10}$.

In Fig. 3.2, we see that our method is able to eventually reach high accuracy even for $\gamma = \frac{1}{30}$, achieving substantial savings in memory/data access. However, there is a tradeoff between memory and computation savings vs. accuracy. Our method is efficient in memory/computation and, at the same time, accurate for $\gamma = \frac{1}{5}$ and $\gamma = \frac{1}{10}$, where it outperforms AK-SVD if the time of each iteration is factored in. We compare with AK-SVD to give an idea of our efficiency, but note that AK-SVD and our CK-SVD are not completely comparable. In our example, both methods reach 100% accuracy eventually but in general they may give different levels of accuracy. The main advantage of CK-SVD appears as the dimensions grow, since then memory/data access is a dominant issue.

Chapter 4

Preconditioned Data Sparsification for Big Data

4.1 Introduction

Principal component analysis (PCA) is a classical unsupervised analysis method commonly used in all quantitative disciplines [78]. 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 chapter 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 [102].

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 [127]. 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 work 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 [7]. 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 4.3. 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., [1, 68, 95]. Many of these schemes are based on multiplying the data on the left by a single random matrix $\mathbf{\Omega}$ and recording $\mathbf{\Omega X}$ 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 $\mathbf{\Omega X}$ are known as sketches or compressive measurements. The motivation behind sketching is the classical Johnson-Lindenstrauss lemma [41] 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 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 [27].

However, for PCA, the desired output is the eigendecomposition of $\mathbf{XX}^T$ or, equiva-

[1] in the sense of [130]

lently, 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 $m' < n$ [68], 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 4.2, summarize important results about the preconditioning transformation and sub-sampling in Section 4.3, provide theoretical results on the sample mean estimator and covariance estimator in sections 4.4 and 4.5 respectively, then focus on results relevant for K-means clustering in Section 4.6 and finish the chapter with numerical experiments in Section 4.7.

4.1.1 Setup

In this chapter, 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., $\|\mathbf{X}\|_{\text{max-row}} = \|\mathbf{X}\|_{2\to\infty} = \sup_{\mathbf{y}\neq 0} \frac{\|\mathbf{X}\mathbf{y}\|_\infty}{\|\mathbf{y}\|_2}$, 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 $X_{i,j}$.

4.1.2 Contributions

In this work, 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 [100]. We present results on the properties of sampling matrices and randomized orthonormal systems in Theorem B.4 of Appendix B and Section 4.3, 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 4.2. Moreover, we

employ recent results on exponential concentration inequalities for matrices [133] to obtain an exponentially decaying bound on the probability that the estimated covariance matrix deviates from its mean value in Theorem 4.3. Numerical simulations are provided to validate the tightness of the concentration bounds derived in this work. 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 4.5. 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.

4.2 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 [42, 60, 106]. For example, Eldar and Gleichman [63] 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 [128, 110, 109, 5] 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 [57, 114, 112, 31, 113, 15]. In particular, Qi and Hughes [114] proposed a method for recovering the mean and principal components of $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ from compressive measurements $\mathbf{R}_i^T\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, \ldots, n$, 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 $\mathbf{P}_i\mathbf{x}_i \in \mathbb{R}^p$, $i = 1, \ldots, n$, 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 $\mathbf{R}_i$, $i = 1, \ldots, n$, is implicitly stored by a seed, the computational cost of the matrix multiplies and inversions is high.

Their work was extended in [113], 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 [87] 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 [114] and [113] 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.

4.2.1 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 [95].[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 [93].

To see this, we recreate the numerical experiment of [93] 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 §4.3 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 [100]: 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 $\operatorname{tr}(\widehat{\mathbf{U}}^T \mathbf{X}\mathbf{X}^T \widehat{\mathbf{U}}) / \operatorname{tr}(\mathbf{X}\mathbf{X}^T)$, and closeness of this fraction to 1 represents

[2] Sampling according to column norm is a common third option, but is generally inferior to leverage-score based sampling

higher accuracy.

Fig. 4.1 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.

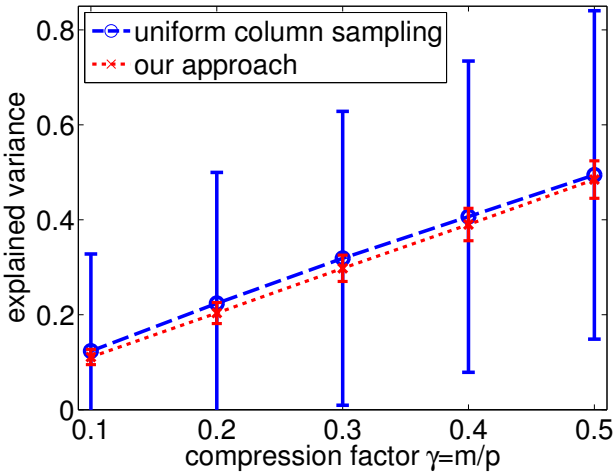


Figure 4.1: Accuracy of estimated PCs via one-pass methods: uniform column sampling and our proposed precondition+sparsification approach. We plot the mean and standard deviation of the explained variance over 1000 runs for each value of γ. The standard deviation for our approach is significantly smaller 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 [47]. 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 [59].

4.2.2 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 [3, 4], 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. [2] 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.

4.2.3 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 [24] is one of the most popular clustering algorithms [145]. 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 [127] and these voluminous data sets make K-means computationally inefficient [26].

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. [55] 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 [32] proposed using approximate SVD algorithms instead.

Other prior works, such as [26, 131], have used randomized schemes for designing efficient clustering algorithms. In particular, Mahoney et al. [26] 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. 4.9g and 4.9i in §4.7. 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 work, 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 [26], 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.

4.3 Preliminaries

The randomized orthonormal system (ROS) is a powerful randomization tool that has found uses in fields from machine learning [7] to numerical linear algebra [68] and compressive sensing [44]. 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 [85, 125].

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

$$\mathbf{x} \mapsto \mathbf{y} = \mathbf{HDx} \tag{4.1}$$

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. 4.1).

We now present relevant results about the preconditioning transformation $\mathbf{HD}$ (4.1) that will be used in the next sections; more sophisticated results are in [132] and Appendix B.4.

Theorem 4.1 (Single element bound for ROS (4.1))**.** *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* $j = 1, \ldots, p$*,*

$$\mathbb{P}\left\{|y_j| \geq (t/\sqrt{p})\|\mathbf{x}\|_2\right\} \leq 2\exp\left(-\eta t^2/2\right) \tag{4.2}$$

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

As pointed out in [132], the proof follows easily from Hoeffding's inequality (Thm. B.1). 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 4.1. *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* $\alpha \in (0, 1)$*,*

$$\mathbb{P}\left\{\|\mathbf{Y}\|_{max} \geq \frac{1}{\sqrt{p}} \cdot \sqrt{\frac{2}{\eta} \log\left(\frac{2np}{\alpha}\right)}\right\} \leq \alpha \tag{4.3}$$

$$\mathbb{P}\left\{\|\mathbf{Y}\|_{max\text{-}row} \geq \sqrt{\frac{n}{p}} \cdot \sqrt{\frac{2}{\eta} \log\left(\frac{2np}{\alpha}\right)}\right\} \leq \alpha \tag{4.4}$$

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 (4.3) 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,

$$\mathbb{P}\left\{\|\mathbf{Y}_{j,:}\|_2 \geq \sqrt{\frac{n}{p}} \cdot t\right\} \leq 2n \exp\left(-\eta t^2/2\right) \tag{4.5}$$

from which (4.4) follows by taking the union bound over p rows and again choosing $t^2 = \frac{2}{\eta}\log(2np/\alpha)$. □

Note that for $\mathbf{H}$ a Hadamard matrix, the result in (4.5) 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 $\mathbf{D} = \mathrm{diag}([D_1, \ldots, D_p])$ 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:

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

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 (4.1) 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 4.1 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 $\mathbf{w} = \mathbf{R}\mathbf{R}^T\mathbf{x}$ 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 (4.1), this almost never happens, and the norm is reduced by nearly m/p as one would hope for:

Corollary 4.2 (Norm bounds for ROS (4.1))**.** *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 $1-\alpha$,*

$$\|\mathbf{w}\|_2^2 \leq \frac{m}{p}\frac{2}{\eta}\log\left(\frac{2p}{\alpha}\right)\|\mathbf{x}\|_2^2 \tag{4.6}$$

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 $1-\alpha$,

$$\|\mathbf{w}_i\|_2^2 \leq \frac{m}{p}\frac{2}{\eta}\log\left(\frac{2np}{\alpha}\right)\|\mathbf{x}_i\|_2^2,\ i = 1, \ldots, n. \tag{4.7}$$

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. 4.1 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. □

As a result of Corollary 4.2, 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.

4.4 The Sample Mean Estimator

We show that a rescaled version of the sample mean of $\left\{\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\right\}_{i=1}^n$, 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 4.2. *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 $\left\{\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\right\}_{i=1}^n$, 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:*

$$\widehat{\overline{\mathbf{x}}}_n = \frac{p}{m}\frac{1}{n}\sum_{i=1}^n \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i. \tag{4.8}$$

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:

$$\tau(m,p) := \max\left\{\left(\frac{p}{m}-1\right),1\right\} = \begin{cases} (\frac{p}{m}-1) & \text{if } \frac{m}{p}\leq 0.5 \\ 1 & \text{if } \frac{m}{p} > 0.5 \end{cases} \tag{4.9}$$

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

$$\delta_1 = 2p\exp\left(\frac{-nt^2/2}{(\frac{p}{m}-1)\|\mathbf{X}\|_{max\text{-}row}^2/n + \tau(m,p)\|\mathbf{X}\|_{max}t/3}\right) \tag{4.10}$$

we have the following ℓ_∞ result:

$$\|\widehat{\overline{\mathbf{x}}}_n - \overline{\mathbf{x}}_n\|_\infty \leq t. \tag{4.11}$$

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. B.4. To upper bound the error, we define:

$$\mathbf{u} := \widehat{\overline{\mathbf{x}}}_n - \overline{\mathbf{x}}_n = \sum_{i=1}^n \frac{1}{n}\left(\frac{p}{m}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \mathbf{x}_i\right) \tag{4.12}$$

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:

$$u_j = \sum_{i=1}^n z_i, \text{ where } z_i = \frac{1}{n}\mathbf{e}_j^T\left(\frac{p}{m}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \mathbf{x}_i\right). \tag{4.13}$$

We now use the Bernstein inequality (Thm. B.2) 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:

$$|z_i| \leq \frac{1}{n} \left\| \frac{p}{m} \mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i - \mathbf{x}_i \right\|_\infty \leq \frac{1}{n} \tau(m,p) \left\| \mathbf{X} \right\|_{\max} \tag{4.14}$$

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. B.4 to compute $\mathbf{\Lambda}$:

$$\begin{aligned} \mathbf{\Lambda} &= \mathbb{E}\Big[\Big(\frac{p}{m} \mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i - \mathbf{x}_i \Big) \Big(\frac{p}{m} \mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i - \mathbf{x}_i \Big)^T \Big] \\ &= \frac{p^2}{m^2} \mathbb{E}\Big[\mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{R}_i \mathbf{R}_i^T \Big] - \mathbf{x}_i \mathbf{x}_i^T \\ &= \frac{-(p-m)}{m(p-1)} \mathbf{x}_i \mathbf{x}_i^T + \frac{p(p-m)}{m(p-1)} \mathrm{diag}\left(\mathbf{x}_i \mathbf{x}_i^T \right). \end{aligned} \tag{4.15}$$

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

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

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

Remark 4.1. *The result in Thm. 4.2 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* (4.10)*, we have:*

$$\frac{1}{\sqrt{p}} \|\widehat{\overline{\mathbf{x}}}_n - \overline{\mathbf{x}}_n\|_2 \leq t. \tag{4.16}$$

Remark 4.2. *Solving for t in* (4.10) *gives the following expression in terms of failure probability δ_1:*

$$\begin{aligned} t = \frac{1}{n} \Bigg\{ & \frac{\tau(m,p)}{3} \|\mathbf{X}\|_{max} \log\big(\frac{2p}{\delta_1}\big) \\ & + \sqrt{\big(\frac{\tau(m,p)}{3} \|\mathbf{X}\|_{max} \log\big(\frac{2p}{\delta_1}\big)\big)^2 + 2\big(\frac{p}{m} - 1\big) \log\big(\frac{2p}{\delta_1}\big) \|\mathbf{X}\|_{max\text{-}row}^2} \Bigg\} \end{aligned} \tag{4.17}$$

We consider a numerical experiment on synthetic data set to show the precision of Thm. 4.2. 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. 4.2, 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. 4.2 reports the average and maximum of 1000 runs for each value of n and compares that with the theoretical error bound t in (4.10) 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.

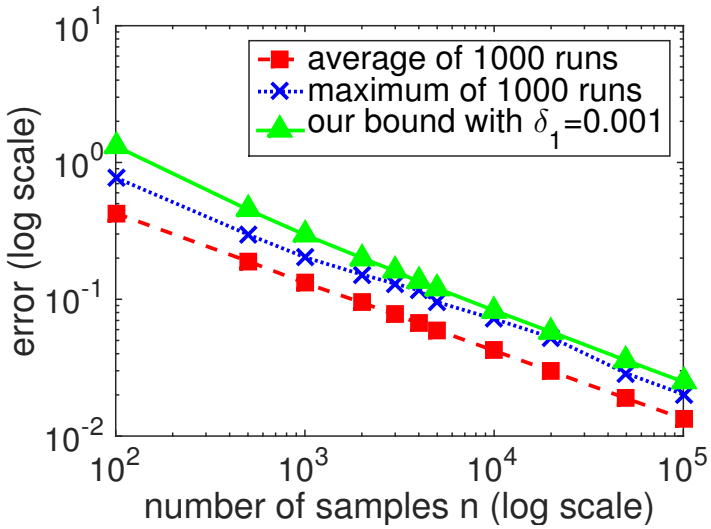


Figure 4.2: Verifying the sharpness of Thm. 4.2 on the synthetic data. For each n we report the average and maximum of the sample mean estimation error in 1000 runs compared with the theoretical error bound when $\delta_1 = 0.001$. The theoretical error bound is tight and decays exponentially as n increases.

In Thm. 4.2, 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 $\|\mathbf{X}\|^2_{\text{max-row}}$ 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 $\|\mathbf{X}\|^2_{\text{max-row}} = n$. Since both $\|\mathbf{X}\|_{\max}$ and $\|\mathbf{X}\|^2_{\text{max-row}}$ 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 4.3 to smooth out large entries of $\mathbf{X}$ and reduce the values of $\|\mathbf{X}\|_{\max}$ and $\|\mathbf{X}\|^2_{\text{max-row}}$.

Corollary 4.3. *In the setting of Thm. 4.2, assume* $\mathbf{X}$ *is preconditioned by the ROS* (4.1) *and* $\gamma = m/p \leq 0.5$. *By using the results of Corollary 4.1, we can find an upper bound for the failure probability* δ_1 *that holds with probability greater than* 0.99*:*

$$\delta_1 \leq 2p \exp\left(\frac{-mnt^2}{4/\eta \log(200np)(1 + \sqrt{p}t/3)}\right), \tag{4.18}$$

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

$$m \geq \frac{1}{n} \cdot \frac{4}{\eta} \log(200np) \log(2000p)(t^{-2} + \frac{\sqrt{p}t^{-1}}{3}) \tag{4.19}$$

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* (4.11). *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* (4.19) *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* $mn \propto \mathcal{O}(\log n)$.

4.5 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 $\left\{\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\right\}_{i=1}^n$, 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 $\left\{\mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i\right\}_{i=1}^n$:

$$\widehat{\mathbf{C}}_{\text{emp}} := \frac{p(p-1)}{m(m-1)} \cdot \frac{1}{n} \sum_{i=1}^n \mathbf{R}_i \mathbf{R}_i^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{R}_i \mathbf{R}_i^T. \tag{4.20}$$

Based on Thm. B.4, we can compute the expectation of $\widehat{\mathbf{C}}_{\text{emp}}$:

$$\mathbb{E}[\widehat{\mathbf{C}}_{\text{emp}}] = \mathbf{C}_{\text{emp}} + \frac{(p-m)}{(m-1)} \operatorname{diag}(\mathbf{C}_{\text{emp}}) \tag{4.21}$$

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 [15], we can easily modify $\widehat{\mathbf{C}}_{\text{emp}}$ to obtain an unbiased estimator:

$$\widehat{\mathbf{C}}_n := \widehat{\mathbf{C}}_{\text{emp}} - \frac{(p-m)}{(p-1)} \operatorname{diag}(\widehat{\mathbf{C}}_{\text{emp}}) \tag{4.22}$$

where revisiting Thm. B.4 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 $\mathbf{HD}$ (4.1) 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 4.2 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 4.3. *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 $\left\{\mathbf{w}_i = \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\right\}_{i=1}^n$, 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:*

$$\widehat{\mathbf{C}}_{emp} = \frac{p(p-1)}{m(m-1)} \cdot \frac{1}{n}\sum_{i=1}^{n} \mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{R}_i\mathbf{R}_i^T. \tag{4.23}$$

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,

$$\widehat{\mathbf{C}}_n = \widehat{\mathbf{C}}_{emp} - \frac{(p-m)}{(p-1)}\operatorname{diag}(\widehat{\mathbf{C}}_{emp}) \tag{4.24}$$

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

$$\mathbb{P}\left\{\|\widehat{\mathbf{C}}_n - \mathbf{C}_{emp}\|_2 \leq t\right\} \geq 1 - \delta_2, \quad \delta_2 = p\exp\left(\frac{-t^2/2}{\sigma^2 + Lt/3}\right) \tag{4.25}$$

where

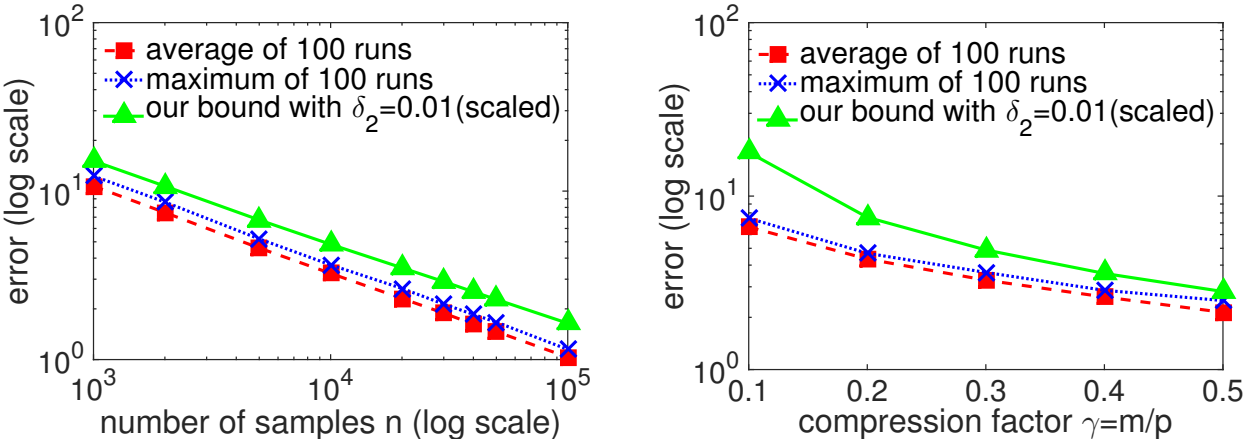
$$L = \frac{1}{n}\left\{\left(\frac{p(p-1)}{m(m-1)}\rho + 1\right)\|\mathbf{X}\|_{max\text{-}col}^2 + \frac{p(p-m)}{m(m-1)}\|\mathbf{X}\|_{max}^2\right\} \tag{4.26}$$

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

$$\begin{aligned}\sigma^2 \leq \frac{1}{n}\Bigg\{&\left(\frac{p(p-1)}{m(m-1)}\rho - 1\right)\|\mathbf{X}\|_{max\text{-}col}^2\|\mathbf{C}_{emp}\|_2 \\ &+ \frac{p(p-1)(p-m)}{m(m-1)^2}\rho\|\mathbf{X}\|_{max\text{-}col}^2\|\operatorname{diag}(\mathbf{C}_{emp})\|_2 \\ &+ \frac{2p(p-1)(p-m)}{m(m-1)^2}\|\mathbf{X}\|_{max}^2\frac{\|\mathbf{X}\|_F^2}{n} \\ &+ \frac{p(p-m)^2}{m(m-1)^2}\frac{\max_{j=1,\ldots,p}\sum_{i=1}^n X_{j,i}^4}{n}\Bigg\}.\end{aligned} \tag{4.27}$$

Proof. The proof follows from the matrix Bernstein inequality (Thm. B.3) and delayed to Appendix B. □

To gain some intuition and verify the accuracy of Thm. 4.3, 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. 4.3(a) shows the closeness of our covariance estimator $\widehat{\mathbf{C}}_n$ to the covariance matrix $\mathbf{C}_{\text{emp}}$, i.e., $\|\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}}\|_2$, 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 $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_5]$ 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 $\|\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}}\|_2$ is compared with the theoretical error bound t in (4.25) when the failure probability is chosen $\delta_2 = 0.01$, and the resulting error bound is scaled by a factor of 10.



(a) (b)

Figure 4.3: Verifying the accuracy of Thm. 4.3 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.

Furthermore, we plot the empirical value of the estimation error vs. compression factor γ in Fig. 4.3(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. 4.3 correctly captures the dependence of the estimation error $\|\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}}\|_2$ 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. 4.3 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 (4.25). Let us assume that $\mathbf{X}$ has normalized columns so that $\|\mathbf{X}\|_{\text{max-col}} = 1$ and $\|\mathbf{X}\|_F^2 = n$. With this normalization, $\|\mathbf{C}_{\text{emp}}\|_2 \leq 1$, and $\|\operatorname{diag}(\mathbf{C}_{\text{emp}})\|_2 \leq 1$ as well, which follows from exercise 27 in §3.3 [70] and $\mathbf{C}_{\text{emp}} \succeq 0$. 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 4.3, 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 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 (4.27) 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

$$\sigma^2 \lesssim \frac{1}{n}\left\{\frac{p}{m}\|\mathbf{C}_{\text{emp}}\|_2 + \frac{p^2}{m^2}\|\operatorname{diag}(\mathbf{C}_{\text{emp}})\|_2 + \frac{2p^2}{m^3} + \frac{p}{m^3}\right\}. \tag{4.28}$$

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 (4.28), we now consider tighter bounds for a 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 $\mathbf{C}_{\text{emp}} = \text{diag}(\mathbf{C}_{\text{emp}}) = p^{-1}\mathbf{I}_p$ in the limit as $n \to \infty$. The third term in (4.28) 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 $\text{diag}(\mathbf{C}_{\text{emp}}) = p^{-1}\mathbf{I}_p$ and thus $\sigma^2 \lesssim 1/n \cdot p/m$, similar to [15]. 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, $\mathbf{C}_{\text{emp}} = \text{diag}(\mathbf{C}_{\text{emp}})$. 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.

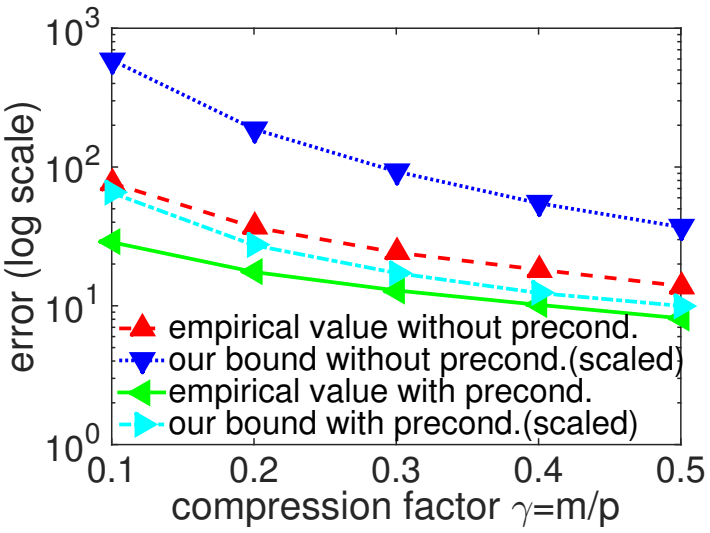


Figure 4.4: Effectiveness of preconditioning on the synthetic data set with few large entries. We plot the average of covariance estimation error over 100 runs for varying γ when $p = 512$ and $n = 1024$ in two cases of sub-sampling $\mathbf{X}$ (without preconditioning) and $\mathbf{Y} = \mathbf{HDX}$ (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.

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. 4.4, 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 4.3, 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. 4.4. 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 4.1. As we see, the 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.

Table 4.1: Number of recovered principal components for various values of compression factor $\gamma = m/p$ over 100 runs

	(without preconditioning)		(with preconditioning)	
γ	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

4.6 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 [24] 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 section, 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:

$$J(\mathbf{c}, \boldsymbol{\mu}) = \sum_{i=1}^{n} \sum_{k=1}^{K} c_{ik} \left\| \mathbf{x}_i - \boldsymbol{\mu}_k \right\|_2^2 . \tag{4.29}$$

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:

$$\forall \;\; i = 1, \ldots, n : \;\; c_{ik} = \begin{cases} 1, & k = \arg\min_j \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|_2^2 \\ 0, & \text{otherwise} \end{cases} \tag{4.30}$$

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

$$\forall \; k = 1, \ldots K : \;\; \boldsymbol{\mu}_k = \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} \mathbf{x}_i \tag{4.31}$$

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 [14] 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.

4.6.1 Optimal objective function via ML estimation

One appealing aspect of the K-means algorithm is that the objective function (4.29) 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:

$$p\left(\mathbf{x}_i|\boldsymbol{\mu}, \mathbf{c}_i\right) = \prod_{k=1}^{K} p\left(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}\right)^{c_{ik}} \tag{4.32}$$

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 $\mathbf{y}_i = \mathbf{HDx}_i$ 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:

$$\boldsymbol{\mu}_k = (\mathbf{HD})^T \boldsymbol{\mu}'_k. \tag{4.33}$$

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. B.4. 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 $\mathbf{z}_i$:

$$p\left(\mathbf{z}_i | \boldsymbol{\mu}', \mathbf{c}_i\right) = \prod_{k=1}^{K} p\left(\mathbf{z}_i | \boldsymbol{\mu}'_k, \boldsymbol{\Sigma}\right)^{c_{ik}}. \tag{4.34}$$

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:

$$J'(\mathbf{c}, \boldsymbol{\mu}') = \sum_{i=1}^{n} \sum_{k=1}^{K} c_{ik} \left\|\mathbf{z}_i - \mathbf{R}_i^T \boldsymbol{\mu}'_k\right\|_2^2. \tag{4.35}$$

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

$$J'(\mathbf{c}, \boldsymbol{\mu}') = \sum_{i=1}^{n} \sum_{k=1}^{K} c_{ik} \left\|\mathbf{R}_i^T \left(\mathbf{y}_i - \boldsymbol{\mu}'_k\right)\right\|_2^2 \tag{4.36}$$

and for $\mathbf{R}_i = \mathbf{I}_p$, $i = 1, \ldots, n$, it reduces to the objective function of the standard K-means (4.29) because the preconditioning transformation $\mathbf{HD}$ is an orthonormal matrix.

4.6.2 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 (4.35), 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:

$$\forall \;\; i = 1, \ldots, n : \;\; c_{ik} = \begin{cases} 1, & k = \arg\min_j \left\| \mathbf{z}_i - \mathbf{R}_i^T \boldsymbol{\mu}'_j \right\|_2^2 \\ 0, & \text{otherwise} \end{cases} \tag{4.37}$$

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 [7], 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 [132] 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. B.5 shows that pairwise distances between each point and cluster centers are preserved.

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

Given the assignments $\mathbf{c}$ from Step 1, we can write (4.35) as:

$$J'(\boldsymbol{\mu}') = \sum_{i \in \mathcal{I}_k} \left\| \mathbf{R}_i^T \left(\mathbf{y}_i - \boldsymbol{\mu}'_k \right) \right\|_2^2 \tag{4.38}$$

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:

$$\left(\sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T \right) \boldsymbol{\mu}'_k = \sum_{i \in \mathcal{I}_k} \mathbf{R}_i \mathbf{R}_i^T \mathbf{y}_i. \tag{4.39}$$

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 (4.39). Given that $n_k^{(j)} > 0$ for all j, $\boldsymbol{\mu}_k'$ is updated as follows:

$$\boldsymbol{\mu}_k' = \operatorname{diag}\left(\left[\frac{1}{n_k^{(1)}},\ldots,\frac{1}{n_k^{(p)}}\right]\right)\left(\sum_{i\in\mathcal{I}_k}\mathbf{R}_i\mathbf{R}_i^T\mathbf{y}_i\right). \tag{4.40}$$

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 4.1.

Algorithm 4.1 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 $\boldsymbol{\mu} = \{\boldsymbol{\mu}_k\}_{k=1}^K \in \mathbb{R}^p$.

function SPARSIFIED K-MEANS($\mathbf{X}, K, \gamma$)
  $\mathbf{Y} \leftarrow \mathbf{HDX}$ ▷ See Eq. (4.1)
  for $i = 1,\ldots,n$ **do**
    $\mathbf{w}_i = \mathbf{R}_i\mathbf{R}_i^T\mathbf{y}_i$ ▷ $\mathbf{R}_i \in \mathbb{R}^{p\times m}$: sampling matrix
  Find initial cluster centers via K-means++ [14]
  for each iteration **do**
    update assignments $\mathbf{c}$ ▷ See Eq. (4.37)
    update cluster centers $\boldsymbol{\mu}'$ ▷ See Eq. (4.40)
  $\boldsymbol{\mu} = (\mathbf{HD})^T\boldsymbol{\mu}'$ ▷ See Eq. (4.33)
  Return $\mathbf{c}$ and $\boldsymbol{\mu}$.

Now, we return to equation (4.39) and study the accuracy of the estimated solution $\boldsymbol{\mu}_k'$. To do this, we re-write (4.39) as:

$$\mathbf{H}_k\boldsymbol{\mu}_k' = \mathbf{m}_k \tag{4.41}$$

where

$$\mathbf{H}_k = \frac{p}{m}\frac{1}{n_k}\sum_{i\in\mathcal{I}_k}\mathbf{R}_i\mathbf{R}_i^T,\ \mathbf{m}_k = \frac{p}{m}\frac{1}{n_k}\sum_{i\in\mathcal{I}_k}\mathbf{R}_i\mathbf{R}_i^T\mathbf{y}_i. \tag{4.42}$$

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 4.4. *Consider* $\mathbf{H}_k$ *defined in* (4.42)*. Then, for all* $t \geq 0$*:*

$$\mathbb{P}\left\{\|\mathbf{H}_k - \mathbf{I}_p\|_2 \leq t\right\} \geq 1 - \delta_3 \tag{4.43}$$

where the failure probability,

$$\delta_3 = p \exp\left(\frac{-n_k t^2/2}{\left(\frac{p}{m} - 1\right) + \left(\frac{p}{m} + 1\right) t/3}\right). \tag{4.44}$$

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. B.4. To apply the matrix Bernstein inequality given in Appendix B, we should find a uniform bound on the spectral norm of each summand $\|\mathbf{Z}_i\|_2$:

$$\|\mathbf{Z}_i\|_2 \leq \frac{1}{n_k}\left(\left\|\frac{p}{m}\mathbf{R}_i\mathbf{R}_i^T\right\|_2 + \|\mathbf{I}_p\|_2\right) = \frac{1}{n_k}\left(\frac{p}{m} + 1\right) \tag{4.45}$$

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. B.4:

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

and thus

$$\sigma^2 = \left\|\sum_{i=1}^{n_k}\mathbb{E}\left[\mathbf{Z}_i^2\right]\right\|_2 = \frac{1}{n_k}\left(\frac{p}{m} - 1\right). \tag{4.46}$$

We now use Theorem B.3 and this completes the proof. □

To verify the accuracy of Thm. 4.4, 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. 4.5. We also compare the empirical values with our theoretical error bound t in (4.44), 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 $\delta_3 = 0.001$.

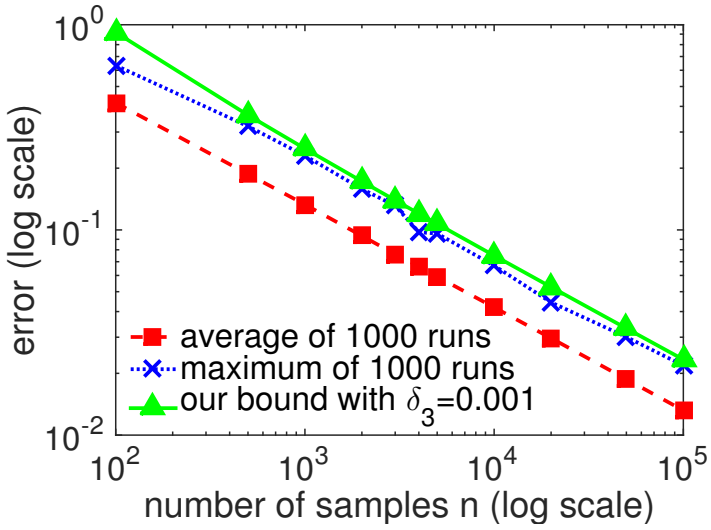


Figure 4.5: Verifying the accuracy of Thm. 4.4. We set parameters $p = 100$ and $\gamma = m/p = 0.3$ and plot the average and maximum of $\|\mathbf{H}_k - \mathbf{I}_p\|_2$ over 1000 runs for varying n. We compare the empirical values with our theoretical bound when $\delta_3 = 0.001$. We see that our bound is tight.

Now, we present a theorem to show the connection between the updated cluster center in our sparsified K-means in (4.41) and the update formula for the standard K-means algorithm.

Theorem 4.5. *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* (4.41). *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* $t \geq 0$*,*

$$\frac{1}{\sqrt{p}}\|\boldsymbol{\mu}_k - \overline{\mathbf{x}}_k\|_2 \leq t\left(1 + \frac{1}{\sqrt{p}}\|\boldsymbol{\mu}_k\|_2\right) \tag{4.47}$$

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

$$\delta_1 = 2p\exp\left(\frac{-n_k t^2/2}{(\frac{p}{m}-1)\|\mathbf{Y}\|^2_{max\text{-}row}/n + \tau(m,p)\|\mathbf{Y}\|_{max}t/3}\right), \tag{4.48}$$

$\tau(m,p)$ *is defined in* (4.9) *and* δ_3 *is given in* (4.44)*. Recall that* $\mathbf{Y} = \mathbf{HDX}$ *is the preconditioned data in our sparsified K-means algorithm.*

Proof. Based on Thm. 4.2 and Thm. 4.4, we re-write equation (4.41) as:

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

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:

$$\begin{aligned}
&\|\boldsymbol{\mu}'_k - \mathbf{HD}\overline{\mathbf{x}}_k\|_2 = \|\mathbf{e} - \mathbf{E}\boldsymbol{\mu}'_k\|_2 \\
&\leq \|\mathbf{e}\|_2 + \|\mathbf{E}\boldsymbol{\mu}'_k\|_2 \leq \|\mathbf{e}\|_2 + \|\mathbf{E}\|_2\|\boldsymbol{\mu}'_k\|_2 \\
&\leq t\sqrt{p}\left(1 + \frac{\|\boldsymbol{\mu}'_k\|_2}{\sqrt{p}}\right)
\end{aligned} \tag{4.49}$$

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. □

4.7 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 [14] for choosing starting points, and re-run the algorithm for different sets of starting points and then choose the results with

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

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 4.2.

Table 4.2: 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

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 4.1 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.

Algorithm 4.2 Sparsified K-means, 2-pass

function SPARSIFIED K-MEANS 2-PASS($\mathbf{X}, K, \gamma$)
  $(\hat{\mathbf{c}}, \hat{\boldsymbol{\mu}})$ =SPARSIFIED K-MEANS($\mathbf{X}, K, \gamma$) ▷ Algorithm 4.1
  for $k = 1, \ldots, K$ **do**
    $\boldsymbol{\mu}_k = \mathbf{0}$, $\mathcal{I}_k = \emptyset$
  for $i = 1, \ldots, n$ **do**
    Find cluster assignment, i.e., k s.t. $\hat{\mathbf{c}}_i = \mathbf{e}_k$
    $\boldsymbol{\mu}_k \mathrel{+}= \mathbf{x}_i$, $\mathcal{I}_k = \mathcal{I}_k \cup \{i\}$
    $\mathbf{c}_i = \arg\min_{k=1,\ldots,K} \|\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k\|_2^2$
  for $k = 1, \ldots, K$ **do**
    $\boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k / |\mathcal{I}_k|$
  Return $\mathbf{c}$ and $\boldsymbol{\mu}$.

Our tests compare with the feature selection and feature extraction algorithms of [26]. In **feature selection**, one first uses a fast approximate SVD algorithm, e.g., [68, 25], 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 $\boldsymbol{\Omega}\mathbf{X}$, where rows of the sampling matrix $\boldsymbol{\Omega} \in \mathbb{R}^{m \times p}$ 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 $\boldsymbol{\Omega}\mathbf{X}$ but the sampling matrix $\boldsymbol{\Omega} \in \mathbb{R}^{m \times p}$ is set to be a random sign matrix. Thus, the computational cost of feature extraction is dominated by the matrix-matrix multiplication $\boldsymbol{\Omega}\mathbf{X}$, whereas the dominant cost in feature selection is the approximate SVD.

4.7.1 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 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. 4.6 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. 4.6 which makes it clear that there is no appreciable difference in quality, while our sparsified K-means algorithm is approximately 67 times faster.

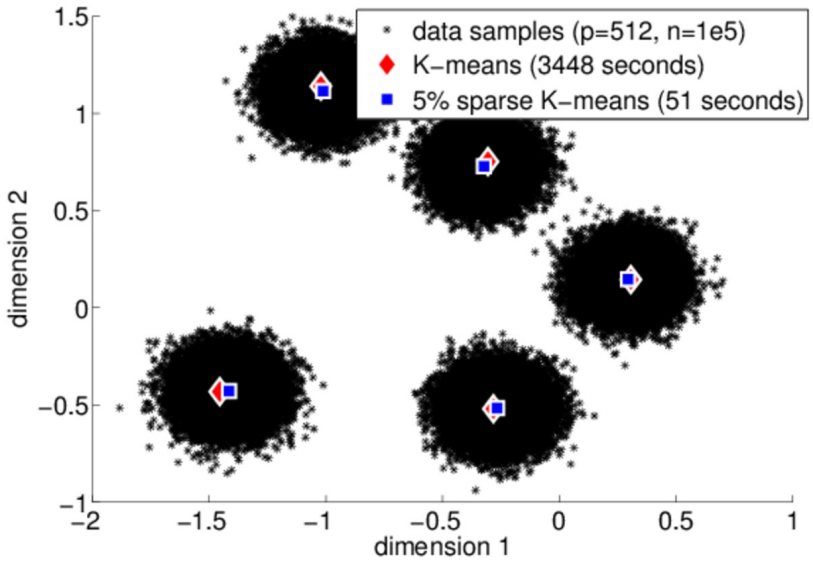


Figure 4.6: Standard K-means and our sparse version of K-means, on synthetic data, $n = 10^5$.

[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 $p \lesssim 20$ [81].

4.7.2 Comparison with dimensionality-reducing approaches on real data

For a realistic clustering application, experiments are performed on the MNIST dataset 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 $n = 21,002$. 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.

Timing and accuracy Clustering accuracy as a function of γ is shown in Fig. 4.7, 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 shown in Fig. 4.8, 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 4.5. 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.

Center estimation The estimated cluster centers $\boldsymbol{\mu}$ from several low-pass K-means algorithms are shown in Fig. 4.9, for $\gamma = 0.03$. As we see, our sparsified K-means algorithm

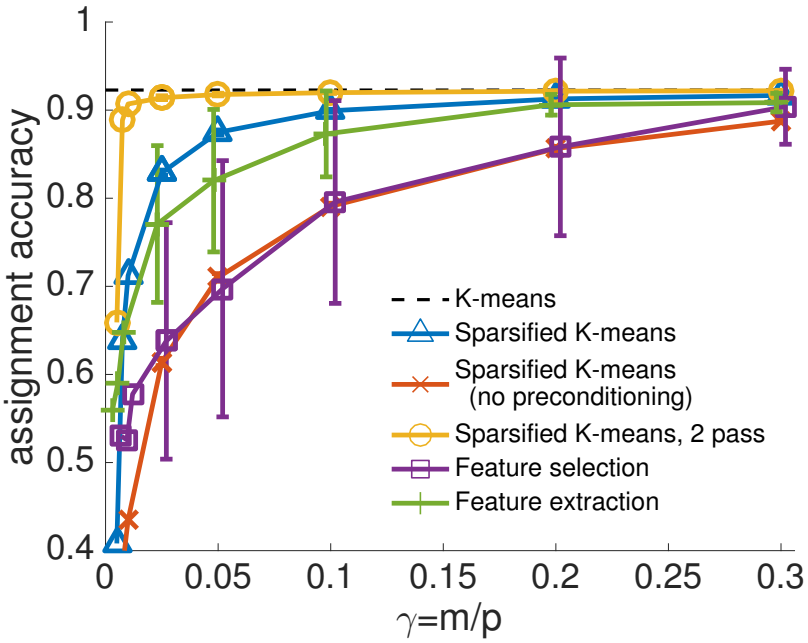


Figure 4.7: Accuracy of various K-means algorithms on the MNIST data, 50 trials. The plot shows the mean and the standard deviation error bars, empirically suggesting that feature-based algorithms show higher variance than the sampling-based algorithm. For example, at $\gamma = 0.1$, the standard deviation 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 $\gamma < 0.025$.

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. 4.4 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

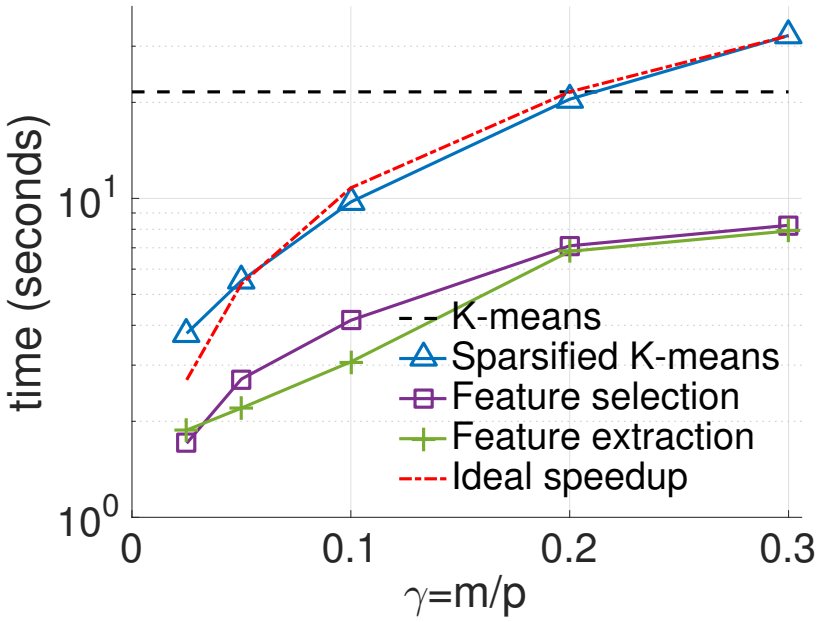


Figure 4.8: Timing of various K-means algorithms on the MNIST data. The three variants of sparsified K-means (with and without preconditioning, and 2-pass) all take approximately the same time on this dataset, so we only show the time for preconditioned sparsified K-means. The red dashed curve is proportional 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.

$n_k \to \infty$ (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 finite first two moments), then from the sampled data we can form the center estimate

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

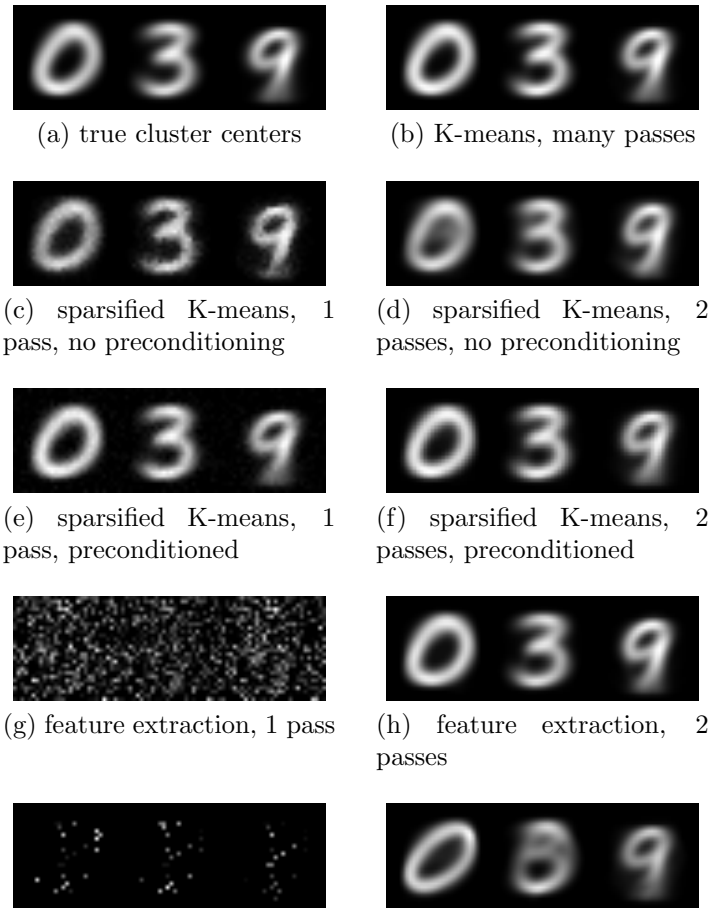
and $\widehat{\boldsymbol{\mu}}_k \to \boldsymbol{\mu}_k$ almost surely as $n_k \to \infty$, which follows from the strong law of large numbers and the independence of the $\mathbf{R}_i$.

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

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

with $\dagger$ representing the pseudo-inverse. As $n_k \to \infty$, this does not converge to $\boldsymbol{\mu}_k$ because

$\mathbf{\Omega}^\dagger\mathbf{\Omega} \neq \mathbf{I}_p$ (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 $\mathbf{\Omega}$ 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.



(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, preconditioned (f) sparsified K-means, 2 passes, preconditioned

(g) feature extraction, 1 pass (h) feature extraction, 2 passes

(i) feature selection, 3 passes (j) feature selection, 4 passes

Figure 4.9: Center estimates $\boldsymbol{\mu}$ from low-pass K-means algorithms.

4.7.3 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. 4.7.

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.

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 [92], using code from the "Infinite MNIST" project[5] , thus $n = 6 \cdot 10^5$, with $p = 784$ 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. 4.10. 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 4.3. 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.

[5] `http://leon.bottou.org/projects/infimnist`

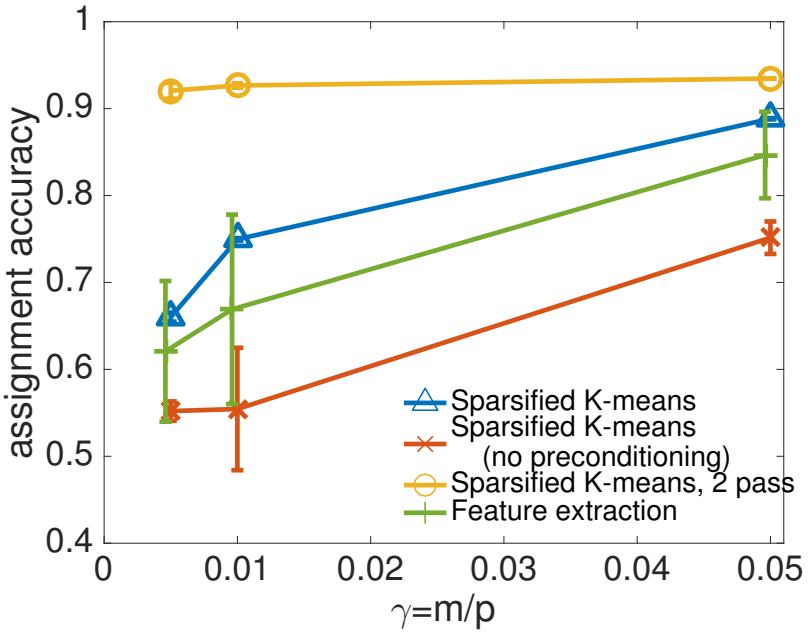


Figure 4.10: Similar to Fig. 4.7 but on much larger data, $n = 6 \cdot 10^5$. The preconditioning helps significantly to increase accuracy, as well as lower variance.

Table 4.3: From the same $n = 6 \cdot 10^5$ simulations as Fig. 4.10, 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

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 4.4 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. 4.2 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.

Table 4.4: 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.

				Time			
	Algorithm	Accuracy	K-means Iterations	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

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 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 4.5, 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.

Table 4.5: Estimated speedup. From the $n = 9{,}631{,}605$ simulation, at $\gamma = 0.05$.

	Time to find assignments		Time to update all centers		Combined time	
Algorithm	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×

4.7.4 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.

Chapter 5

Randomized Clustered Nyström for Large-Scale Kernel Machines

5.1 Introduction

Kernel machines have been widely used in various machine learning problems such as classification, clustering, and regression. In kernel-based learning, the input data points are mapped to a high-dimensional feature space and the pairwise inner products in the lifted space are computed and stored in a positive semidefinite kernel matrix $\mathbf{K}$. The lifted representation may lead to better performance of the learning problem, but a drawback is the need to store and manipulate a large kernel matrix of size $n \times n$, where n is the size of data set. Thus a kernel machine has quadratic space complexity and quadratic or cubic computational complexity (depending on the specific type of machine).

One promising strategy for reducing these costs consists of a low-rank approximation of the kernel matrix $\mathbf{K} \approx \mathbf{L}\mathbf{L}^T$, where $\mathbf{L} \in \mathbb{R}^{n \times r}$ for a target rank $r < n$. Such low-rank approximations can be used to reduce the memory and computation cost by trading-off accuracy for scalability. For this reason, much research has focused on efficient algorithms for computing low-rank approximations, e.g., [56, 16, 17, 68]. The Nyström method is probably one of the most well-studied and successful methods that has been used to scale up several kernel methods [84, 129]. The Nyström method works by selecting a small set of bases referred to as "landmark points" and computing the kernel similarities between the input data points and landmark points. Therefore, the performance of the Nyström method depends crucially on the number of selected landmark points as well as the procedure

according to which these landmark points are selected.

The original Nyström method, first introduced to the kernel machine setting by [143], proposed to select landmark points uniformly at random from the set of input data points. More recently, several other probabilistic strategies have been proposed to provide informative landmark points in the Nyström method, including sampling with weights proportional to column norms [46], diagonal entries [48], and leverage scores [62]. The authors in [147] proposed a non-probabilistic technique for generating landmark points using centroids resulting from K-means clustering on the input data points. The proposed "Clustered Nyström method" shows the Nyström approximation error is related to the encoding power of landmark points in summarizing data and it provides improved accuracy over other sampling methods such as uniform and column-norm sampling [83]. However, the main drawback of this method is the high memory and computational complexity associated with performing K-means clustering on high-dimensional large-scale data sets.

The aim of this work is to improve the accuracy and efficiency of the Nyström method in two directions. We present a novel algorithm to compute the optimal rank-r approximation in the Nyström method when the number of landmark points exceed the rank parameter r. In fact, our proposed method can be used within all landmark selection procedures to compute the best rank-r approximation achievable by a chosen set of landmark points. Moreover, we present an efficient method for landmark selection which provides a tunable tradeoff between the accuracy of low-rank approximations and memory/computation requirements. Our proposed "Randomized Clustered Nyström method" generates a set of landmark points based on low-dimensional random projections of the input data points [1].

In more detail, our main contributions are threefold.

- It is common to select more landmark points than the target rank r to obtain high quality Nyström low-rank approximations. In Section 5.4, we present a novel algorithm with theoretical analysis for computing the optimal rank-r approximation

when the number of landmark points exceed the target rank r. Thus, our proposed method, called "Nyström via QR Decomposition," can be used with any landmark selection algorithm to find the best rank-r approximation for a given set of landmark points. We also provide intuitive and real-world examples to show the superior performance and efficiency of our method in Section 5.4.

- Second, we present a random-projection-type landmark selection algorithm which easily scales to large-scale high-dimensional data sets. Our proposed "Randomized Clustered Nyström method" presented in Section 5.5 performs the K-means clustering algorithm on the random projections of input data points and it requires only two passes over the original data set. Thus our method leads to significant memory and computation savings in comparison with the Clustered Nyström method. Moreover, our theoretical results (Theorem 5.1) show that the proposed method produces low-rank approximations with little loss in accuracy compared to Clustered Nyström with high probability.

- Third, we present extensive numerical experiments comparing our Randomized Clustered Nyström method with a few other sampling methods on two tasks: (1) low-rank approximation of kernel matrices and (2) kernel ridge regression. In Section 5.6, we consider six data sets from the LIBSVM archive [29] with dimensionality up to $p = 150{,}360$.

5.2 Notation and Preliminaries

We denote column vectors with lower-case bold letters and matrices with upper-case bold letters. $\mathbf{I}_{n\times n}$ is the identity matrix of size $n \times n$; $\mathbf{0}_{m\times n}$ is the $m \times n$ matrix of zeros. For a vector $\mathbf{x} \in \mathbb{R}^p$, let $\|\mathbf{x}\|_2$ denote the Euclidean norm, and $\mathrm{diag}(\mathbf{x})$ represents a diagonal matrix with the elements of $\mathbf{x}$ on the main diagonal. The Frobenius norm for a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ is defined as $\|\mathbf{A}\|_F = (\sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}^2)^{1/2} = (\mathrm{tr}(\mathbf{A}^T\mathbf{A}))^{1/2}$, where A_{ij} represents the (i,j)-th

entry of $\mathbf{A}$, $\mathbf{A}^T$ is the transpose of $\mathbf{A}$, and $\text{tr}(\cdot)$ is the trace operator.

Let $\mathbf{K} \in \mathbb{R}^{n \times n}$ be a symmetric positive semidefinite (SPSD) matrix with $\text{rank}(\mathbf{K}) = \rho \leq n$. The singular value decomposition (SVD) or eigenvalue decomposition of $\mathbf{K}$ can be written as $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{U} \in \mathbb{R}^{n \times \rho}$ contains the orthonormal eigenvectors, i.e., $\mathbf{U}^T\mathbf{U} = \mathbf{I}_{\rho \times \rho}$, and $\mathbf{\Lambda} = \text{diag}\left([\lambda_1(\mathbf{K}), \ldots, \lambda_\rho(\mathbf{K})]\right) \in \mathbb{R}^{\rho \times \rho}$ is a diagonal matrix which contains the eigenvalues of $\mathbf{K}$ in descending order, i.e., $\lambda_1(\mathbf{K}) \geq \ldots \geq \lambda_\rho(\mathbf{K})$. The matrices $\mathbf{U}$ and $\mathbf{\Lambda}$ can be decomposed for a target rank r ($r \leq \rho$):

$$\begin{aligned} \mathbf{K} &= \begin{pmatrix} \mathbf{U}_r & \mathbf{U}_{\rho-r} \end{pmatrix} \begin{pmatrix} \mathbf{\Lambda}_r & \mathbf{0}_{r \times (\rho-r)} \\ \mathbf{0}_{(\rho-r) \times r} & \mathbf{\Lambda}_{\rho-r} \end{pmatrix} \begin{pmatrix} \mathbf{U}_r^T \\ \mathbf{U}_{\rho-r}^T \end{pmatrix} \\ &= \mathbf{U}_r\mathbf{\Lambda}_r\mathbf{U}_r^T + \mathbf{U}_{\rho-r}\mathbf{\Lambda}_{\rho-r}\mathbf{U}_{\rho-r}^T, \end{aligned} \tag{5.1}$$

where $\mathbf{\Lambda}_r \in \mathbb{R}^{r \times r}$ contains the r leading eigenvalues and the columns of $\mathbf{U}_r \in \mathbb{R}^{n \times r}$ span the top r-dimensional eigenspace, and $\mathbf{\Lambda}_{\rho-r}$ and $\mathbf{U}_{\rho-r}$ contain the remaining $(\rho - r)$ eigenvalues and eigenvectors. It is well-known that $\mathbf{K}_{(r)} = \mathbf{U}_r\mathbf{\Lambda}_r\mathbf{U}_r^T$ is the "best rank-r approximation" to $\mathbf{K}$ in the sense that $\mathbf{K}_{(r)}$ minimizes $\|\mathbf{K} - \mathbf{A}\|_F$ over all matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$ of rank at most r [49] and we have $\|\mathbf{K} - \mathbf{K}_{(r)}\|_F = (\sum_{i=r+1}^{\rho} \lambda_i(\mathbf{K})^2)^{1/2}$. If $\lambda_r(\mathbf{K}) = \lambda_{r+1}(\mathbf{K})$, then $\mathbf{K}_{(r)}$ is not unique, so we write $\mathbf{K}_{(r)}$ to mean any matrix satisfying Equation 5.1. The pseudo-inverse of $\mathbf{K}$ can be obtained from the SVD or eigenvalue decomposition as $\mathbf{K}^\dagger = \mathbf{U}_\rho\mathbf{\Lambda}_\rho^{-1}\mathbf{U}_\rho^T$. When $\mathbf{K}$ is full rank, we have $\mathbf{K}^\dagger = \mathbf{K}^{-1}$.

Another matrix factorization technique that we use in this work is the QR decomposition. An $n \times m$ matrix $\mathbf{A}$, with $n \geq m$, can be decomposed as a product of two matrices $\mathbf{A} = \mathbf{Q}\mathbf{R}$, where $\mathbf{Q} \in \mathbb{R}^{n \times m}$ has m orthonormal columns, i.e., $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_{m \times m}$, and $\mathbf{R} \in \mathbb{R}^{m \times m}$ is an upper triangular matrix. Sometimes this is called the thin QR decomposition, to distinguish it from a full QR decomposition which finds $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and zero-pads $\mathbf{R}$ accordingly.

5.3 Background and Related Work

Kernel methods have been successfully applied to a variety of machine learning problems such as classification and regression. Well-known examples include support vector machines (SVM) [38], kernel principal component analysis (KPCA) [121], kernel ridge regression [122], kernel clustering [61], and kernel dictionary learning [138]. The main idea behind kernel-based learning is to map the input data points into a feature space, where all pairwise inner products of the mapped data points can be computed via a nonlinear kernel function that satisfies Mercer's condition [11, 120]. Thus, kernel methods allow one to use linear algorithms in the higher (or infinite) dimensional feature space which correspond to nonlinear algorithms in the original space. For this reason, kernel machines have received much attention as an effective tool to tackle problems with complex and nonlinear structures.

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ be a data matrix that contains n data points in $\mathbb{R}^p$ as its columns. The inner products in feature space are calculated using a "kernel function" $\kappa(\cdot, \cdot)$ defined on the original space:

$$K_{ij} := \kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \quad \forall i, j = 1, \ldots, n,$$

where $\Phi : \mathbf{x} \mapsto \Phi(\mathbf{x})$ is the kernel-induced feature map. All pairwise inner products of the n mapped data points are stored in the so-called "kernel matrix" $\mathbf{K} \in \mathbb{R}^{n \times n}$, where the (i, j)-th entry is K_{ij}. Two well-known examples of kernel functions that lead to symmetric positive semidefinite (SPSD) kernel matrices are Gaussian and polynomial kernel functions. The former takes the form $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / c)$ and the polynomial kernel is of the form $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$, where $c \in \mathbb{R}^+$ and $d \in \mathbb{N}$ are the parameters [137, 111]. Moreover, combinations of multiple kernels can be constructed to tackle problems with complex and heterogeneous data sources [18, 66, 90].

Despite the simplicity of kernel machines in nonlinear representation of data, one prominent problem is the calculation, storage, and manipulation of the kernel matrix for large-scale data sets. The cost to form $\mathbf{K}$ using standard kernel functions is $\mathcal{O}(pn^2)$ and it takes $\mathcal{O}(n^2)$

memory to store the full kernel matrix. Thus, both memory and computation cost scale as the square of the number of data points. Moreover, subsequent processing of the kernel matrix within the learning process is computationally quite expensive. For example, algorithms such as KPCA and kernel dictionary learning compute the eigenvalue decomposition of the kernel matrix, where the standard techniques take $\mathcal{O}(n^3)$ time and multiple passes over $\mathbf{K}$ will be required. In other kernel-based learning methods such as kernel ridge regression, the inverse of the kernel matrix $(\mathbf{K} + \lambda \mathbf{I}_{n\times n})^{-1}$, where $\lambda > 0$ is a regularization parameter, must be computed which requires $\mathcal{O}(n^3)$ time [37, 9]. Thus, large-scale data sets have provided a considerable challenge to the design of efficient kernel-based learning algorithms [127, 71].

A well-studied approach to reduce the memory and computation burden associated with kernel machines is to use a low-rank approximation of kernel matrices. This approach utilizes the decaying spectra of kernel matrices and the best rank-r approximation $\mathbf{K}_{(r)} = \mathbf{U}_r \mathbf{\Lambda}_r \mathbf{U}_r^T$ is computed, cf. Equation 5.1. Since $\mathbf{K}$ is SPSD, the eigenvalue decomposition can be used to express a low-rank approximation in the form of:

$$\mathbf{K}_{(r)} = \mathbf{L}\mathbf{L}^T, \;\; \mathbf{L} = \mathbf{U}_r \mathbf{\Lambda}_r^{1/2} \in \mathbb{R}^{n\times r}.$$

The benefits of this low-rank approximation are twofold. First, it takes $\mathcal{O}(nr)$ to store the matrix $\mathbf{L}$ which is only linear in the data set size n. The reduction of memory requirements from quadratic to linear results in significant memory savings. Second, the low-rank approximation leads to substantial computational savings within the learning process. For example, the following matrix inversion arising in algorithms such as kernel ridge regression can be calculated using the Sherman-Morrison-Woodbury formula:

$$\begin{aligned}
(\mathbf{K} + \lambda \mathbf{I}_{n\times n})^{-1} &\approx \left(\mathbf{K}_{(r)} + \lambda \mathbf{I}_{n\times n}\right)^{-1} \\
&= \left(\mathbf{L}\mathbf{L}^T + \lambda \mathbf{I}_{n\times n}\right)^{-1} \\
&= \lambda^{-1}\left(\mathbf{I}_{n\times n} - \mathbf{L}\left(\mathbf{L}^T\mathbf{L} + \lambda \mathbf{I}_{r\times r}\right)^{-1}\mathbf{L}^T\right). \qquad (5.2)
\end{aligned}$$

Here, we only need to invert a much smaller matrix of size just $r \times r$. Thus, the computation cost is $\mathcal{O}(nr^2 + r^3)$ to compute $\mathbf{L}^T\mathbf{L}$ and the matrix inversion in Equation 5.2.

Another example of computation savings is the "linearization" of kernel methods using the low-rank approximation, where linear algorithms are applied to the rows of $\mathbf{L} \in \mathbb{R}^{n \times r}$. In this case, the matrix $\mathbf{L}$ serves as an empirical kernel map and the rows of $\mathbf{L}$ are known as virtual samples. This strategy has been shown to speed up various kernel-based learning methods such as SVM, kernel dictionary learning, and kernel clustering [148, 64, 108].

While the low-rank approximation of kernel matrices is a promising approach to reduce the memory and computational complexity, the main bottleneck is the computation of the full kernel matrix $\mathbf{K}$ and the best rank-r approximation $\mathbf{K}_{(r)}$. Standard algorithms for computing the eigenvalue decomposition of $\mathbf{K}$ take $\mathcal{O}(n^3)$ time. Partial eigenvalue decomposition, e.g., Krylov subspace method, can be performed to find the r leading eigenvalues/eigenvectors. However, these techniques require at least r passes over the entire kernel matrix which is prohibitive for large dense matrices [68].

To address this problem, much recent work has focused on efficient randomized methods to compute low-rank approximations of large matrices [95]. The Nyström method is one of the few randomized approximation techniques that does not need to first compute the entire kernel matrix. The standard Nyström method was first introduced (in the context of matrix kernel approximation) in [143] and is based on sampling a small subset of input data columns, after which the kernel similarities between the small subset and input data points are computed to construct a rank-r approximation. Section 5.3.1 discusses in detail the Nyström method and its extension which finds the approximate eigenvalue decomposition of the kernel matrix.

Since the sampling technique is a key aspect of the Nyström method, much research has focused on selecting the most informative subset of input data to improve the approximation accuracy and thus the performance of kernel-based learning methods [83]. An overview of different sampling techniques, including the Clustered Nyström method, is presented in Section 5.3.2.

5.3.1 The Nyström method

The Nyström method for generating a low-rank approximation of the SPSD kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ works by selecting a small set of bases referred to as "landmark points". For example, the simplest and most common technique to select the landmark points is based on uniform sampling without replacement from the set of all input data points [143]. In this section, we explain the Nyström method for a given set of landmark points regardless of the sampling mechanism.

Let $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_m] \in \mathbb{R}^{p \times m}$ be the set of m landmark points in $\mathbb{R}^p$. The Nyström method first constructs two matrices $\mathbf{C} \in \mathbb{R}^{n \times m}$ and $\mathbf{W} \in \mathbb{R}^{m \times m}$, where $C_{ij} = \kappa(\mathbf{x}_i, \mathbf{z}_j)$ and $W_{ij} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$. Next, it uses both $\mathbf{C}$ and $\mathbf{W}$ to construct a low-rank approximation of the kernel matrix $\mathbf{K}$:

$$\mathbf{G} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T.$$

For the rank-restricted case, the Nyström method generates a rank-r approximation of the kernel matrix, $r \leq m$, by computing the best rank-r approximation of the $m \times m$ inner matrix $\mathbf{W}$ [83, 84, 129, 86, 140]:

$$\mathbf{G}^{nys}_{(r)} = \mathbf{C}\mathbf{W}^{\dagger}_{(r)}\mathbf{C}^T, \tag{5.3}$$

where $\mathbf{W}^{\dagger}_{(r)}$ represents the pseudo-inverse of $\mathbf{W}_{(r)}$. Thus, the eigenvalue decomposition of the matrix $\mathbf{W}$ should be computed to find the top r eigenvalues and corresponding eigenvectors. Let $\mathbf{\Sigma}_r \in \mathbb{R}^{r \times r}$ and $\mathbf{V}_r \in \mathbb{R}^{m \times r}$ contain the top r eigenvalues and the corresponding orthonormal eigenvectors of $\mathbf{W}$, respectively. Then, the rank-r approximation in Equation 5.3 can be expressed as:

$$\mathbf{G}^{nys}_{(r)} = \mathbf{L}^{nys}\left(\mathbf{L}^{nys}\right)^T, \;\; \mathbf{L}^{nys} = \mathbf{C}\mathbf{V}_r\left(\mathbf{\Sigma}^{\dagger}_r\right)^{1/2} \in \mathbb{R}^{n \times r}. \tag{5.4}$$

The time complexity of the Nyström method to form $\mathbf{L}^{nys}$ is $\mathcal{O}(pnm + m^2r + nmr)$, where it takes $\mathcal{O}(pnm)$ to construct matrices $\mathbf{C}$ and $\mathbf{W}$. Also, it takes $\mathcal{O}(m^2r)$ time to perform the

partial eigenvalue decomposition of $\mathbf{W}$ and $\mathcal{O}(nmr)$ represents the cost of matrix multiplication $\mathbf{C}\mathbf{V}_r$. Thus, for $r \leq m \ll n$, the computation cost to form the low-rank approximation of the kernel matrix, $\mathbf{K} \approx \mathbf{L}^{nys}(\mathbf{L}^{nys})^T$, is only linear in the data set size n.

In practice, there exist two approaches to obtain the approximate eigenvalue decomposition of the kernel matrix in the Nyström method. The first approach is based on the exact eigenvalue decomposition of $\mathbf{W}$ to get the following estimates of the r leading eigenvalues and eigenvectors of $\mathbf{K}$ [83]:

$$\widehat{\mathbf{U}}_r^{(1)} = \sqrt{\frac{m}{n}}\mathbf{C}\mathbf{V}_r\mathbf{\Sigma}_r^{\dagger}, \;\; \widehat{\mathbf{\Lambda}}_r^{(1)} = \frac{n}{m}\mathbf{\Sigma}_r. \tag{5.5}$$

These estimates of eigenvalues/eigenvectors are naive since it is easy to show that the estimated eigenvectors are not guaranteed to be orthonormal, i.e., $(\widehat{\mathbf{U}}_r^{(1)})^T\widehat{\mathbf{U}}_r^{(1)} \neq \mathbf{I}_{r\times r}$. Moreover, the factor n/m in Equation 5.5 is used to roughly compensate for the small size of the matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$ compared to the $n \times n$ kernel matrix. Thus, the accuracy of this approach depends heavily on the data set and the selected landmark points.

The second approach provides more accurate estimates of eigenvalues and eigenvectors of $\mathbf{K}$ by using the low-rank approximation in Equation 5.4, and in fact this approach provides the exact eigenvalue decomposition of $\mathbf{G}_{(r)}^{nys}$. The first step is to find the exact eigenvalue decomposition of the $r \times r$ matrix:

$$\left(\mathbf{L}^{nys}\right)^T\mathbf{L}^{nys} = \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T,$$

where $\tilde{\mathbf{V}}, \tilde{\mathbf{\Sigma}} \in \mathbb{R}^{r\times r}$. Then, the estimates of r leading eigenvalues and eigenvectors of $\mathbf{K}$ are obtained as follows [147]:

$$\widehat{\mathbf{U}}_r^{(2)} = \mathbf{L}^{nys}\tilde{\mathbf{V}}\left(\tilde{\mathbf{\Sigma}}^{\dagger}\right)^{1/2}, \;\; \widehat{\mathbf{\Lambda}}_r^{(2)} = \tilde{\mathbf{\Sigma}}.$$

For this case, the resultant eigenvectors are orthonormal:

$$\begin{aligned}\left(\widehat{\mathbf{U}}_r^{(2)}\right)^T\widehat{\mathbf{U}}_r^{(2)} &= \left(\tilde{\mathbf{\Sigma}}^{\dagger}\right)^{1/2}\widetilde{\mathbf{V}}^T\left(\mathbf{L}^{nys}\right)^T\mathbf{L}^{nys}\widetilde{\mathbf{V}}\left(\tilde{\mathbf{\Sigma}}^{\dagger}\right)^{1/2} \\ &= \left(\tilde{\mathbf{\Sigma}}^{\dagger}\right)^{1/2}\left(\widetilde{\mathbf{V}}^T\widetilde{\mathbf{V}}\right)\tilde{\mathbf{\Sigma}}\left(\widetilde{\mathbf{V}}^T\widetilde{\mathbf{V}}\right)\left(\tilde{\mathbf{\Sigma}}^{\dagger}\right)^{1/2} = \mathbf{I}_{r\times r},\end{aligned}$$

Algorithm 5.1 Standard Nyström

Input: data set $\mathbf{X}$, landmark points $\mathbf{Z}$, kernel function κ, target rank r
Output: estimates of r leading eigenvectors and eigenvalues of the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$: $\widehat{\mathbf{U}}_r^{(2)} \in \mathbb{R}^{n \times r}$, $\widehat{\mathbf{\Lambda}}_r^{(2)} \in \mathbb{R}^{r \times r}$

1: Form two matrices $\mathbf{C}$ and $\mathbf{W}$: $C_{ij} = \kappa(\mathbf{x}_i, \mathbf{z}_j)$, $W_{ij} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$
2: Compute the eigenvalue decomposition: $\mathbf{W} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T$
3: Form the matrix: $\mathbf{L}^{nys} = \mathbf{C}\mathbf{V}_r \left(\mathbf{\Sigma}_r^{\dagger}\right)^{1/2}$
4: Compute the eigenvalue decomposition: $(\mathbf{L}^{nys})^T\mathbf{L}^{nys} = \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T$
5: $\widehat{\mathbf{U}}_r^{(2)} = \mathbf{L}^{nys}\tilde{\mathbf{V}}\left(\tilde{\mathbf{\Sigma}}^{\dagger}\right)^{1/2}$ and $\widehat{\mathbf{\Lambda}}_r^{(2)} = \tilde{\mathbf{\Sigma}}$

where this comes from the fact that $\widetilde{\mathbf{V}}$ contains orthonormal eigenvectors and $(\widetilde{\mathbf{\Sigma}}^{\dagger})^{1/2}\widetilde{\mathbf{\Sigma}}(\widetilde{\mathbf{\Sigma}}^{\dagger})^{1/2} = \mathbf{I}_{r \times r}$. The overall procedure to estimate the r leading eigenvalues/eigenvectors based on the Nyström method is summarized in Algorithm 5.1. The time complexity of the approximate eigenvalue decomposition is $\mathcal{O}(nr^2 + r^3)$, in addition to the cost of computing $\mathbf{L}^{nys}$ mentioned earlier.

5.3.2 Sampling techniques for the Nyström method

The importance of landmark points in the Nyström method has driven much recent work into various probabilistic and deterministic sampling techniques to improve the accuracy of Nyström-based approximations [83, 129]. In this section, we review a few popular sampling methods in the literature.

The simplest and most common sampling method proposed originally by [143] was uniform sampling without replacement. In this case, each data point in the data set is sampled with the same probability, i.e., $p_i = \frac{1}{n}$, for $i = 1, \ldots, n$. The advantage of this technique is the low computational complexity associated with sampling landmark points. However, it has been shown that uniform sampling does not take into account the nonuniform structure of many data sets. Therefore, sampling mechanisms based on nonuniform distributions have been proposed to address this problem. Two such examples include: (1) "Column-norm sampling" [46], where m columns of the kernel matrix are sampled with weights proportional to the ℓ_2 norm of columns of $\mathbf{K}$ (**not** of the data matrix $\mathbf{X}$), i.e., $p_i = \|\mathbf{k}_i\|_2^2 / \|\mathbf{K}\|_F^2$,

and (2) "diagonal sampling" [48], where the weights are proportional to the corresponding diagonal elements, i.e., $p_i = K_{ii}^2 / \sum_{i=1}^{n} K_{ii}^2$. The former requires $\mathcal{O}(n^2)$ time and space to find the nonuniform distribution, while the latter requires $\mathcal{O}(n)$ time and space. The column-norm sampling method requires computing the entire kernel matrix $\mathbf{K}$, which negates one of the principal benefits of the Nyström method. The diagonal sampling method reduces to the uniform sampling for shift-invariant kernels, such as the Gaussian kernel function, since $K_{ii} = 1$ for all $i = 1, \ldots, n$. Recently, [62] have studied both empirical and theoretical aspects of uniform and nonuniform sampling on the accuracy of Nyström-based low-rank approximations.

The "Clustered Nyström method" proposed by [147, 149] is a popular non-probabilistic approach that uses out-of-sample extensions to select informative landmark points. The key observation of their work is that the Nyström low-rank approximation error depends on the quantization error of encoding the entire data set with the landmark points. For this reason, the Clustered Nyström method sets the landmark points to be the centroids found from K-means clustering. In machine learning and pattern recognition, K-means clustering [24] is a well-established technique to partition a data set into clusters by trying to minimize the total sum of the squared Euclidean distances of each point to the closest cluster center.

To present the main result of Clustered Nyström method, we first explain K-means clustering briefly. Given $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$, an m-partition of this data set is a collection $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ of m disjoint and nonempty sets (each representing a cluster) such that their union covers the entire data set. Each cluster can be defined by a cluster center, which is the sample mean of data points in that cluster. Thus, the goal of K-means clustering is to minimize the following:

$$E\left(\mathbf{X}, \mathcal{S}\right) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mu(\mathbf{x}_i)\|_2^2,$$

where $\mu(\mathbf{x}_i) \in \mathbb{R}^p$ represents the centroid of the cluster to which the data point $\mathbf{x}_i$ is assigned, and hence depends on $\mathcal{S}$. The optimal clustering $\mathcal{S}^{opt}$ is the solution of following NP-hard

optimization problem [24]:

$$\mathcal{S}^{opt} = \arg\min_{\mathcal{S}} E\left(\mathbf{X}, \mathcal{S}\right). \tag{5.6}$$

In practice, Lloyd's algorithm [91], also known as the K-means clustering algorithm, is used to solve the optimization problem in Equation 5.6. The K-means clustering algorithm is an iterative procedure which consists of two steps: (1) data points are assigned to the nearest cluster centers, and (2) the cluster centers are updated based on the most recent assignment of the data points. The objective function decreases at every step, and so the procedure is guaranteed to terminate since there are only finitely many partitions. Typically, only a few iterations are needed to converge to a locally optimal solution. The quality of clustering can be improved by using well-chosen initialization, such as K-means++ initialization [14].

Now, we present the result of the Clustered Nyström method which relates the Nyström approximation error (in terms of the Frobenius norm) to the quantization error induced by encoding the data set with landmark points [147].

Proposition 5.1 (Clustered Nyström Method)**.** *Assume that the kernel function κ satisfies the following property:*

$$\left(\kappa(\mathbf{a}, \mathbf{b}) - \kappa(\mathbf{c}, \mathbf{d})\right)^2 \leq \eta \left(\|\mathbf{a} - \mathbf{c}\|_2^2 + \|\mathbf{b} - \mathbf{d}\|_2^2\right), \quad \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{R}^p \tag{5.7}$$

where η is a constant depending on κ. Consider the data set $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ and the landmark set $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_m] \in \mathbb{R}^{p \times m}$ which partitions the data set $\mathbf{X}$ into m clusters $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$. Let $\mu(\mathbf{x}_i)$ denote the closest landmark point to each data point $\mathbf{x}_i$:

$$\mu\left(\mathbf{x}_i\right) = \arg\min_{\mathbf{z}_j \in \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}} \|\mathbf{x}_i - \mathbf{z}_j\|_2.$$

Consider the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, and the Nyström approximation $\mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T$, where $C_{ij} = \kappa(\mathbf{x}_i, \mathbf{z}_j)$ and $W_{ij} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$. The approximation error in terms of the Frobenius norm is upper bounded:

$$\mathcal{E} = \|\mathbf{K} - \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T\|_F \leq \eta_1 \sqrt{E\left(\mathbf{X}, \mathcal{S}\right)} + \eta_2 E\left(\mathbf{X}, \mathcal{S}\right) \tag{5.8}$$

where η_1 and η_2 are two constants and $E(\mathbf{X}, \mathcal{S})$ is the total quantization error of encoding each data point $\mathbf{x}_i$ with the closest landmark point $\mu(\mathbf{x}_i)$:

$$E\left(\mathbf{X}, \mathcal{S}\right) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mu\left(\mathbf{x}_i\right)\|_2^2. \tag{5.9}$$

In [147], it is shown that for a number of widely used kernel functions, e.g., linear, polynomial, and Gaussian, the property in Equation 5.7 is satisfied. Based on Proposition 5.1, the Clustered Nyström method tries to minimize the total quantization error in Equation 5.9—and thus the Nyström approximation error—by performing the K-means algorithm on the n data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. The resulting m cluster centers are then chosen as the landmark points to construct matrices $\mathbf{C}$ and $\mathbf{W}$ and generate the low-rank approximation $\mathbf{G} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T$. One benefit of the approach is that the full kernel matrix $\mathbf{K}$ is never formed.

5.4 Improved Nyström Approximation via QR Decomposition

In Section 5.3.1, we explained the Nyström method to compute rank-r approximations of SPSD kernel matrices based on a set of landmark points. For a data set of size n and a small set of m landmark points ($m \geq r$), two matrices $\mathbf{C} \in \mathbb{R}^{n \times m}$ and $\mathbf{W} \in \mathbb{R}^{m \times m}$ are constructed to form the low-rank approximation of $\mathbf{K} \in \mathbb{R}^{n \times n}$: $\mathbf{G} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T$, where $\text{rank}(\mathbf{G}) \leq m$.

Although the final goal is to find an approximation that has rank no greater than r, it is often preferred to select $m > r$ landmark points and then restrict the resultant approximation to have rank at most r, e.g., [129, 86, 140]. The main intuition is that selecting $m > r$ landmark points and then restricting the approximation to a lower rank-r space has a regularization effect which can lead to more accurate approximations [62]. For example, Proposition 5.1 states that the approximation error $\|\mathbf{K} - \mathbf{G}\|_F$ is a function of the total quantization error induced by encoding data points with the set of m landmark points. Obviously, the more landmark points are selected, the total quantization error becomes smaller and thus the quality of rank-r approximation can be improved. Therefore, it is

important to use an efficient and accurate method to restrict the matrix $\mathbf{G}$ to have rank at most r.

In the standard Nyström method presented in Algorithm 5.1, the rank of matrix $\mathbf{G}$ is restricted by computing the best rank-r approximation of the inner matrix $\mathbf{W}$: $\mathbf{G}_{(r)}^{nys} = \mathbf{C}\mathbf{W}_{(r)}^{\dagger}\mathbf{C}^T$. Since the inner matrix in the representation of $\mathbf{G}_{(r)}^{nys}$ has rank no greater than r, it follows that $\mathbf{G}_{(r)}^{nys}$ has rank at most r. The main benefit of this technique is the low computational cost of performing an exact eigenvalue decomposition or SVD on a relatively small matrix of size $m \times m$. However, the standard Nyström method totally ignores the structure of the matrix $\mathbf{C}$ and is solely based on "filtering" $\mathbf{W}$. In fact, since the rank-r approximation $\mathbf{G}_{(r)}^{nys}$ does not utilize the full knowledge of matrix $\mathbf{C}$, the selection of more landmark points does not guarantee an improved low-rank approximation in the standard Nyström method.

To solve this problem, we present an efficient method to compute the best rank-r approximation of the matrix $\mathbf{G} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T$, for given matrices $\mathbf{C} \in \mathbb{R}^{n\times m}$ and $\mathbf{W} \in \mathbb{R}^{m\times m}$. In contrast with the standard Nyström method, our proposed approach takes advantage of both matrices $\mathbf{C}$ and $\mathbf{W}$. To begin, let us consider the best rank-r approximation of the matrix $\mathbf{G}$:

$$\begin{aligned}
\mathbf{G}_{(r)}^{opt} &= \underset{\mathbf{G}' \text{: } \mathrm{rank}(\mathbf{G}')\leq r}{\arg\min} \|\mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T - \mathbf{G}'\|_F \\
&\overset{(a)}{=} \underset{\mathbf{G}' \text{: } \mathrm{rank}(\mathbf{G}')\leq r}{\arg\min} \|\mathbf{Q}\underbrace{\mathbf{R}\mathbf{W}^{\dagger}\mathbf{R}^T}_{m\times m}\mathbf{Q}^T - \mathbf{G}'\|_F \\
&\overset{(b)}{=} \underset{\mathbf{G}' \text{: } \mathrm{rank}(\mathbf{G}')\leq r}{\arg\min} \|\left(\mathbf{Q}\mathbf{V}'\right)\mathbf{\Sigma}'\left(\mathbf{Q}\mathbf{V}'\right)^T - \mathbf{G}'\|_F \\
&= \left(\mathbf{Q}\mathbf{V}'_r\right)\mathbf{\Sigma}'_r\left(\mathbf{Q}\mathbf{V}'_r\right)^T, \qquad (5.10)
\end{aligned}$$

where (a) follows from the QR decomposition of $\mathbf{C} \in \mathbb{R}^{n\times m}$; $\mathbf{C} = \mathbf{Q}\mathbf{R}$, where $\mathbf{Q} \in \mathbb{R}^{n\times m}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$. To get (b), the eigenvalue decomposition of the $m \times m$ matrix $\mathbf{R}\mathbf{W}^{\dagger}\mathbf{R}^T$ is computed, $\mathbf{R}\mathbf{W}^{\dagger}\mathbf{R}^T = \mathbf{V}'\mathbf{\Sigma}'\mathbf{V}'^T$, where the diagonal matrix $\mathbf{\Sigma}' \in \mathbb{R}^{m\times m}$ contains m eigenvalues in descending order on the main diagonal and the columns of $\mathbf{V}' \in \mathbb{R}^{m\times m}$ are

Algorithm 5.2 Nyström via QR Decomposition

Input: data set $\mathbf{X}$, landmark points $\mathbf{Z}$, kernel function κ, target rank r
Output: estimates of r leading eigenvectors and eigenvalues of the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$: $\widehat{\mathbf{U}}_r^{opt} \in \mathbb{R}^{n \times r}$, $\widehat{\mathbf{\Lambda}}_r^{opt} \in \mathbb{R}^{r \times r}$

1: Form two matrices $\mathbf{C}$ and $\mathbf{W}$: $C_{ij} = \kappa(\mathbf{x}_i, \mathbf{z}_j)$, $W_{ij} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$
2: Perform the QR decomposition: $\mathbf{C} = \mathbf{QR}$
3: Compute the eigenvalue decomposition: $\mathbf{RW}^{\dagger}\mathbf{R}^T = \mathbf{V}'\mathbf{\Sigma}'\mathbf{V}'^T$
4: $\widehat{\mathbf{U}}_r^{opt} = \mathbf{QV}'_r$ and $\widehat{\mathbf{\Lambda}}_r^{opt} = \mathbf{\Sigma}'_r$

the corresponding eigenvectors. Moreover, we note that the columns of $\mathbf{QV}' \in \mathbb{R}^{n \times m}$ are orthonormal because both $\mathbf{Q}$ and $\mathbf{V}'$ have orthonormal columns:

$$\big(\mathbf{QV}'\big)^T\big(\mathbf{QV}'\big) = \mathbf{V}'^T\big(\mathbf{Q}^T\mathbf{Q}\big)\mathbf{V}' = \mathbf{V}'^T\mathbf{V}' = \mathbf{I}_{m \times m}.$$

Thus, the decomposition $(\mathbf{QV}')\mathbf{\Sigma}'(\mathbf{QV}')^T$ contains the m eigenvalues and orthonormal eigenvectors of the Nyström approximation $\mathbf{CW}^{\dagger}\mathbf{C}^T$. Based on the Eckart-Young theorem, the best rank-r approximation of $\mathbf{G} = \mathbf{CW}^{\dagger}\mathbf{C}^T$ is then computed using the r leading eigenvalues $\mathbf{\Sigma}'_r \in \mathbb{R}^{r \times r}$ and corresponding eigenvectors $\mathbf{QV}'_r \in \mathbb{R}^{n \times r}$, as given in Equation 5.10. Thus, the estimates of the top r eigenvalues and eigenvectors of the kernel matrix $\mathbf{K}$ from the Nyström approximation $\mathbf{CW}^{\dagger}\mathbf{C}^T$ are obtained as follows:

$$\widehat{\mathbf{U}}_r^{opt} = \mathbf{QV}'_r, \;\; \widehat{\mathbf{\Lambda}}_r^{opt} = \mathbf{\Sigma}'_r. \tag{5.11}$$

These estimates can also be used to approximate the kernel matrix as $\mathbf{K} \approx \mathbf{L}^{opt}\left(\mathbf{L}^{opt}\right)^T$, where $\mathbf{L}^{opt} = \widehat{\mathbf{U}}_r^{opt}\big(\widehat{\mathbf{\Lambda}}_r^{opt}\big)^{1/2}$.

The overall procedure to estimate the r leading eigenvalues/eigenvectors of the kernel matrix $\mathbf{K}$ based on a set of landmark points $\mathbf{Z} \in \mathbb{R}^{p \times m}$, $m \geq r$, is presented in Algorithm 5.2. The time complexity of this method is $\mathcal{O}(pnm + nm^2 + m^3 + nmr)$, where $\mathcal{O}(pnm)$ represents the cost to form matrices $\mathbf{C}$ and $\mathbf{W}$. The complexity of the QR decomposition is $\mathcal{O}(nm^2)$ and it takes $\mathcal{O}(m^3)$ time to compute the eigenvalue decomposition of $\mathbf{RW}^{\dagger}\mathbf{R}^T$. Finally, the cost to compute the matrix multiplication $\mathbf{QV}'_r$ is $\mathcal{O}(nmr)$.

We can compare the computational complexity of our proposed Nyström method via QR decomposition (Algorithm 5.2) with that of the standard Nyström method (Algorithm

5.1). Since our focus in this work is on large-scale data sets with n large, we only consider terms involving n which lead to dominant computation costs. Based on the discussion in Section 5.3.1, it takes $\mathcal{C}_{nys} = \mathcal{O}(pnm + nmr + nr^2)$ time to compute the eigenvalue decomposition using the standard Nyström method. For our proposed method, the cost of eigenvalue decomposition is $\mathcal{C}_{opt} = \mathcal{O}(pnm + nmr + nm^2)$. Thus, for data of even moderate dimension with $p \gtrsim m$, the dominant term in both $\mathcal{C}_{nys}$ and $\mathcal{C}_{opt}$ is $\mathcal{O}(pnm)$. This means that the increase in computation cost of our method (nm^2 vs. nr^2) becomes less significant when the number of landmark points m is close to the target rank r.

In the rest of this section, we compare the performance and efficiency of our proposed method presented in Algorithm 5.2 with Algorithm 5.1 on three examples. As we will see, our proposed method yields more accurate decompositions than the standard Nyström method for small values of m, such as $m = 2r$.

5.4.1 Toy example

It is always true that for any kernel matrix $\mathbf{K}$, $\|\mathbf{K} - \mathbf{G}_{(r)}^{opt}\|_F \leq \|\mathbf{K} - \mathbf{G}_{(r)}^{nys}\|_F$ (this is also true in the spectral norm), due to the best-approximation properties of our estimator. We can show, using examples, that this inequality can be quite loose.

In the first example, we consider a small kernel matrix of size 3×3:

$$\mathbf{K} = \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1.01 & 0 \\ 10 & 0 & 100 \end{bmatrix}.$$

Such a matrix could arise, for example, using the polynomial kernel with parameters $c = 0$ and $d = 1$ and the data matrix:

$$\mathbf{X} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 10 \\ 0 & \sqrt{2 \cdot 1.01} & 0 \\ 1 & 0 & 10 \end{bmatrix}.$$

Here, the goal is to compute the rank $r = 1$ approximation of $\mathbf{K}$. Suppose that $m = 2$ columns of the kernel matrix are sampled uniformly, e.g., the first and second columns. Then, we have:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1.01 \\ 10 & 0 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} 1 & 0 \\ 0 & 1.01 \end{bmatrix}.$$

In the standard Nyström method, the best rank-1 approximation of the inner matrix $\mathbf{W}$ is first computed.[1] Then, based on Equation 5.3, the rank-1 approximation of the kernel matrix in the standard Nyström method is given by:

$$\mathbf{G}_{(1)}^{nys} = \begin{bmatrix} 1 & 0 \\ 0 & 1.01 \\ 10 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{1.01} \end{bmatrix} \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1.01 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1.01 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The normalized kernel approximation error in terms of the Frobenius norm is large: $\|\mathbf{K} - \mathbf{G}_{(1)}^{nys}\|_F / \|\mathbf{K}\|_F = 0.99$. On the other hand, using the same matrices $\mathbf{C}$ and $\mathbf{W}$, our proposed method first computes the QR decomposition of $\mathbf{C} = \mathbf{QR}$:

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{\sqrt{101}} & 0 \\ 0 & 1 \\ \frac{10}{\sqrt{101}} & 0 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \sqrt{101} & 0 \\ 0 & 1.01 \end{bmatrix}.$$

Then, the product of three matrices $\mathbf{RW}^{\dagger}\mathbf{R}^T$ is computed to find its eigenvalue decomposition

[1] One might ask if it is better to first find $\mathbf{W}^{\dagger}$ and then find the best rank-r approximation of $\mathbf{W}^{\dagger}$. This generally does not help, and one can construct similar toy examples where this approach does arbitrarily poorly as well.

$\mathbf{R}\mathbf{W}^{\dagger}\mathbf{R}^{T} = \mathbf{V}'\mathbf{\Sigma}'\mathbf{V}'^{T}$:

$$\begin{aligned}
\mathbf{R}\mathbf{W}^{\dagger}\mathbf{R}^{T} &= \begin{bmatrix} \sqrt{101} & 0 \\ 0 & 1.01 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{1.01} \end{bmatrix} \begin{bmatrix} \sqrt{101} & 0 \\ 0 & 1.01 \end{bmatrix} \\
&= \begin{bmatrix} 101 & 0 \\ 0 & 1.01 \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{V}'} \underbrace{\begin{bmatrix} 101 & 0 \\ 0 & 1.01 \end{bmatrix}}_{\mathbf{\Sigma}'} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{V}'^{T}}.
\end{aligned}$$

Finally, the rank-1 approximation of the kernel matrix in our proposed method is obtained by using Equation 5.10:

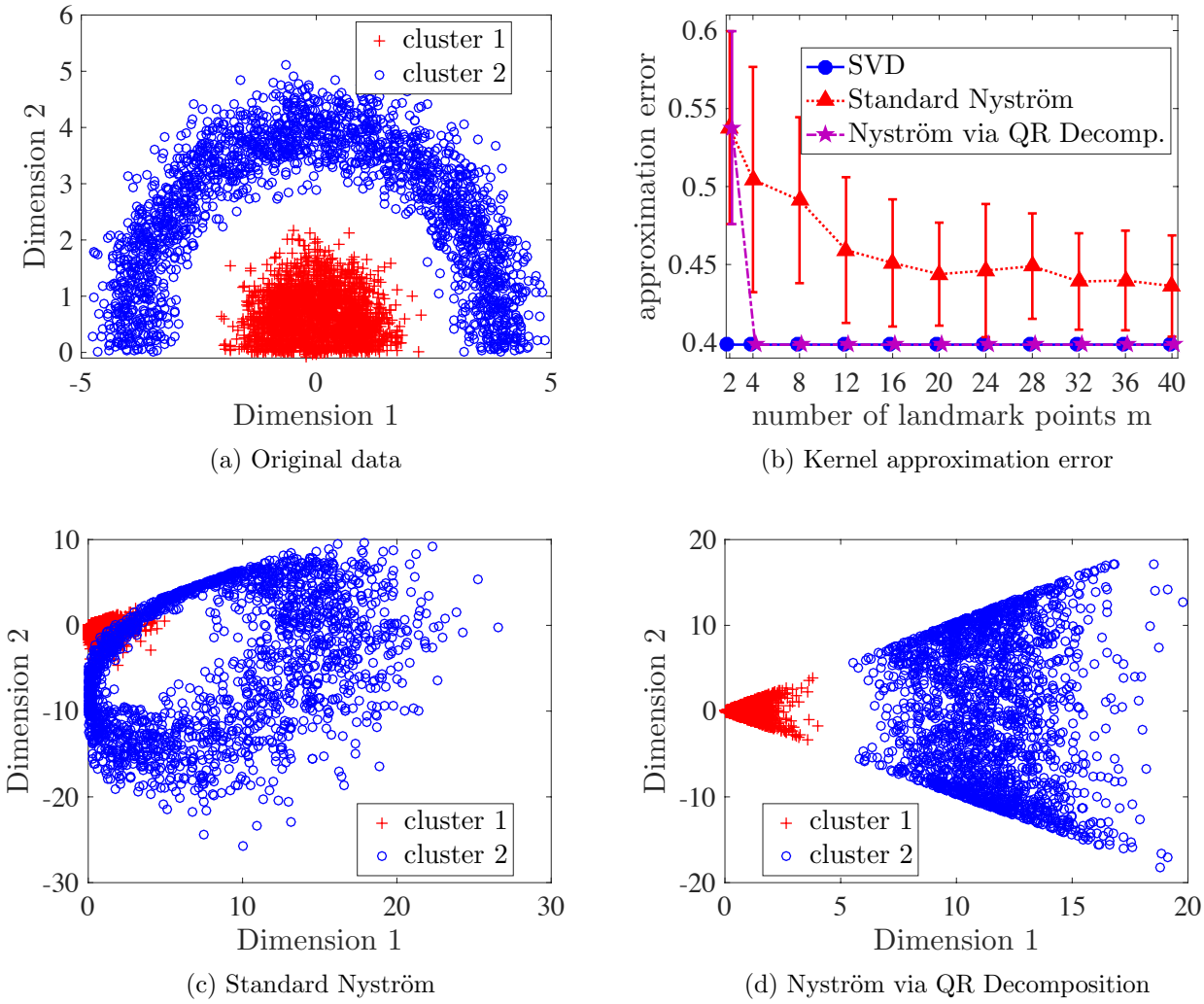
$$\mathbf{G}_{(1)}^{opt} = \begin{bmatrix} \frac{1}{\sqrt{101}} & 0 \\ 0 & 1 \\ \frac{10}{\sqrt{101}} & 0 \end{bmatrix} \begin{bmatrix} 101 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{101}} & 0 & \frac{10}{\sqrt{101}} \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 10 \\ 0 & 0 & 0 \\ 10 & 0 & 100 \end{bmatrix},$$

where $\|\mathbf{K} - \mathbf{G}_{(1)}^{opt}\|_F / \|\mathbf{K}\|_F = 0.01$. In fact, one can show that our approximation is the same as the best rank-1 approximation formed using full knowledge of $\mathbf{K}$, i.e., $\mathbf{G}_{(1)}^{opt} = \mathbf{K}_{(1)}$. Furthermore, clearly we can tweak this toy example to make the error $\|\mathbf{K} - \mathbf{G}_{(1)}^{opt}\|_F / \|\mathbf{K}\|_F = \epsilon$ and $\|\mathbf{K} - \mathbf{G}_{(1)}^{nys}\|_F / \|\mathbf{K}\|_F = 1 - \epsilon$ for any $\epsilon > 0$. This example demonstrates that Nyström via QR Decomposition produces a much more accurate rank-1 approximation of the kernel matrix with the same matrices $\mathbf{C}$ and $\mathbf{W}$ used in the standard Nyström method.

5.4.2 Synthetic data set

As shown in Figure 5.1a, we consider a synthetic data set consisting of $n = 4000$ data points in $\mathbb{R}^2$ that are nonlinearly separable. Therefore, a nonlinear kernel function is employed to find an embedding of these points so that linear learning algorithms can be applied to the mapped data points. To do this, we use the polynomial kernel function with the degree $d = 2$ and the constant $c = 0$, i.e., $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2$. Next, a low-rank

approximation of the kernel matrix in the form of $\mathbf{K} \approx \mathbf{L}\mathbf{L}^T$, $\mathbf{L} \in \mathbb{R}^{n \times r}$, is computed by using the Nyström method. The n rows of $\mathbf{L}$ represent the virtual samples or mapped data points [148, 64, 108]. Given a suitable kernel function and accurate low-rank approximation technique, the n rows of $\mathbf{L}$ in $\mathbb{R}^r$ are linearly separable. In this example, we set the target rank $r = 2$ so that we can easily visualize the resultant mappings.



(a) Original data

(b) Kernel approximation error

(c) Standard Nyström

(d) Nyström via QR Decomposition

Figure 5.1: The standard Nyström method is compared with our proposed "Nyström via QR Decomposition" on the synthetic data set ($p = 2$, $n = 4000$). The polynomial kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2$ is used to find nonlinear mappings of the original data points by using the rank-2 approximation of the kernel matrix $\mathbf{K} \approx \mathbf{L}\mathbf{L}^T$, $\mathbf{L} \in \mathbb{R}^{n \times 2}$. The bottom row uses $m = 4$ landmark points.

We measure the approximation accuracy by using the normalized kernel approximation error defined as $\|\mathbf{K} - \mathbf{L}\mathbf{L}^T\|_F / \|\mathbf{K}\|_F$, where the matrix $\mathbf{L}$ is obtained by using the standard Nyström method and our proposed method "Nyström via QR Decomposition". In Figure 5.1b, the mean and standard deviation of the normalized kernel approximation error over 50 trials for varying number of landmark points m are reported. In each trial, the m landmark points are chosen uniformly at random without replacement from the input data. Both our method and the standard Nyström method share same matrices $\mathbf{C}$ and $\mathbf{W}$ for a fair comparison. As we expect, the accuracy of our Nyström via QR decomposition is exactly the same as the standard Nyström method for $m = r = 2$. As the number of landmark points m increases, the accuracy of standard Nyström method improves and it slowly gets closer to the accuracy of exact eigenvalue decomposition or SVD. However, our proposed method reaches the accuracy of SVD even for $m = 2r = 4$. In fact, we observe that the approximation error of our method by using $m = 4$ landmark points is better than the accuracy of standard Nyström method with $m = 40$. For this example, our proposed rank-r approximation technique in Algorithm 5.2 is more accurate and memory efficient than the standard Nyström method with at least one order of magnitude savings in memory.

Finally, we visualize the mapped data points using both methods for fixed $m = 4$. In Figure 5.1c and Figure 5.1d, the rows of $\mathbf{L}^{nys} \in \mathbb{R}^{n\times 2}$ and $\mathbf{L}^{opt} \in \mathbb{R}^{n\times 2}$ are plotted, respectively. The rows of $\mathbf{L}^{opt} \in \mathbb{R}^{n\times 2}$ in the "Nyström via QR Decomposition" method are linearly separable which is desirable for kernel-based learning. But, the rows of $\mathbf{L}^{nys} \in \mathbb{R}^{n\times 2}$ are not linearly separable due to the poor performance of the standard Nyström method.

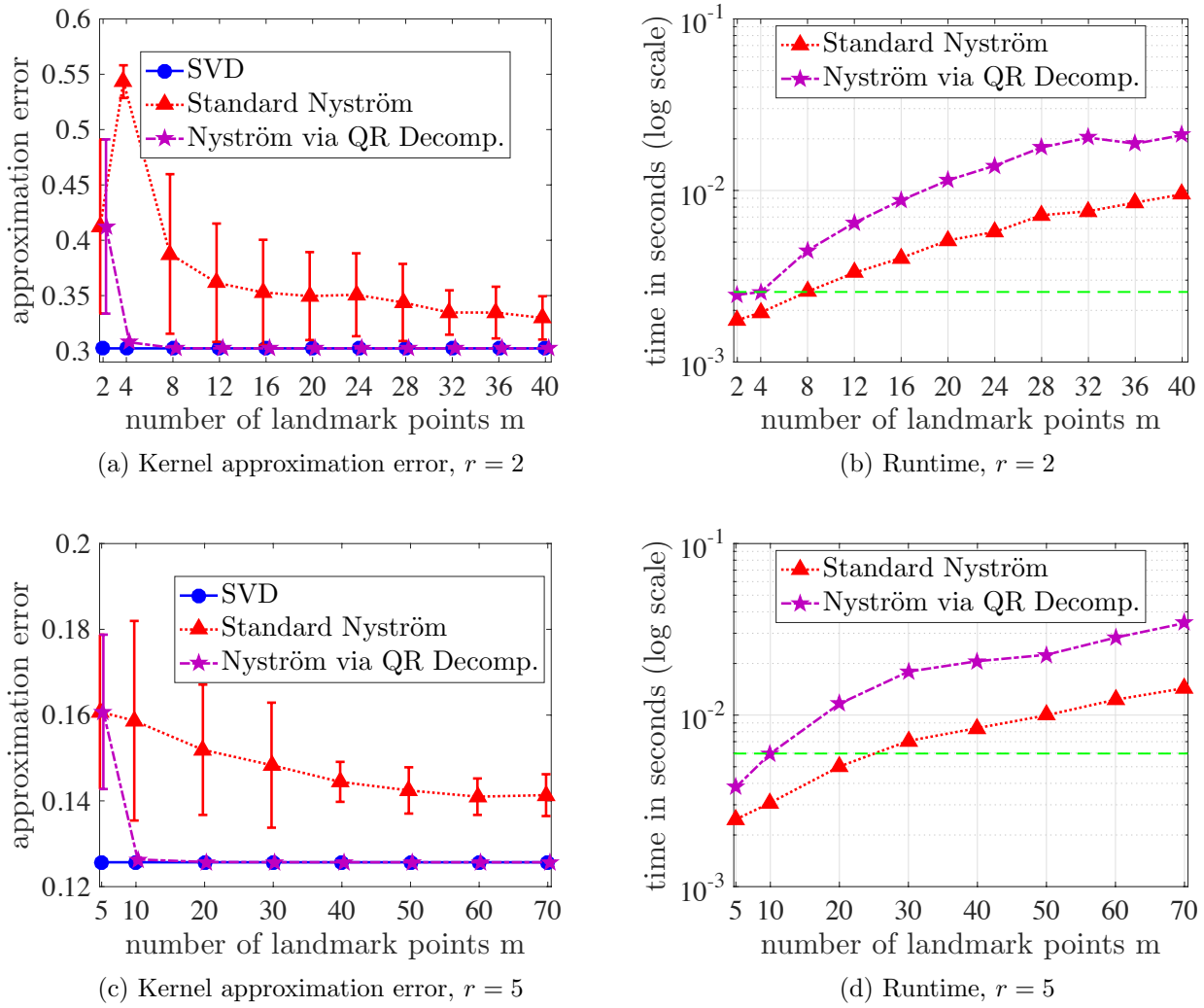
5.4.3 Real data set

In the last example, we use the `satimage` data set [29] with $p = 36$ and $n = 4435$. We duplicate each data point four times to increase to $n = 17{,}740$ in order to have a more meaningful comparison of computation times. The kernel matrix is formed using the Gaussian kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / c\right)$ where the parameter c is chosen

as the averaged squared distance between all the data points and the sample mean [147]. The m landmark points are chosen by performing K-means on the original data, following the Clustered Nyström method.

In Figure 5.2a and Figure 5.2c, the mean and standard deviation of normalized kernel approximation error are reported over 50 trials for varying number of landmark points m and two values of the target rank $r = 2$ and $r = 5$, respectively. As expected, when the number of landmark points is set to be the same as the target rank, the standard Nyström method and our proposed method have exactly the same approximation error. Interestingly, it is seen that when the number of landmark points m increases, the approximation error does not necessarily decrease in the standard Nyström method as shown in Figure 5.2a. This is a major drawback of the standard Nyström method because the increase in memory and computation costs imposed by larger m may lead to worse performance. In contrast, our proposed "Nyström via QR Decomposition" outperforms the standard Nyström method for both values of the target rank $r = 2$ and $r = 5$, and we know theoretically that performance can only improve as m increases. Moreover, we see that the accuracy of our method reaches the accuracy of the best rank-r approximation obtained by using the SVD for as few as $m = 2r$ landmark points.

The runtime of both methods are also compared in Figure 5.2b and Figure 5.2d for two cases of $r = 2$ and $r = 5$, respectively. The reported values are averaged over 50 trials and they represent the computation cost associated with Algorithm 5.1 and Algorithm 5.2. As we explained earlier in this section, the computational complexity of our method $\mathcal{C}_{opt}$ will be slightly increased compared to the standard Nyström method $\mathcal{C}_{nys}$ and this is consistent with the timing results in Figure 5.2b and Figure 5.2d. Moreover, we see that the runtime of our method is increased by almost a factor of 2 even for large values of m. To have a fair comparison, we draw a dashed green line that determines the values of m for which both methods have the same running time. In Figure 5.2b, the runtime for $m = 4$ in our method is the same as $m = 8$ in the standard Nyström, while our method is much more accurate.



(a) Kernel approximation error, $r = 2$

(b) Runtime, $r = 2$

(c) Kernel approximation error, $r = 5$

(d) Runtime, $r = 5$

Figure 5.2: The standard Nyström method is compared with our proposed "Nyström via QR Decomposition" on the satimage data set. Our method yields more accurate low-rank approximations with noticeable memory and computation savings.

Similarly, in Figure 5.2d, the runtime for $m = 10$ in our method is almost the same as $m = 30$ in the standard Nyström method. However, our method results in more accurate low-rank approximation of the kernel matrix. This dataset further supports that our "Nyström via QR Decomposition" results in more accurate low-rank approximations than the standard Nyström method with significant memory and computation savings.

5.5 Randomized Clustered Nyström Method

The selection of informative landmark points is an essential component to obtain accurate low-rank approximations of SPSD matrices in the Nyström method. The Clustered Nyström method [147] has been shown to be a powerful technique for generating highly accurate low-rank approximations compared to uniform sampling and other sampling methods [83, 129, 73]. However, the main drawbacks of this method are high memory and computational complexities associated with performing K-means clustering on large-scale data sets. In this section, we introduce an efficient randomized method for generating a set of representative landmark points based on low-dimensional random projections of the original data. Specifically, our proposed method provides a "tunable tradeoff" between the accuracy of Nyström low-rank approximations and the efficiency in terms of memory and computation savings.

To introduce our proposed method, we begin by explaining the process of generating landmark points in the Clustered Nyström method. As mentioned in Section 5.3.2, the central idea behind Clustered Nyström is that the approximation error depends on the total quantization error of encoding each data point in the data set with the closest landmark point. Thus, landmark points are chosen to be centroids resulting from the K-means clustering algorithm which partitions the data set into m clusters. Given an initial set of m centroids $\{\boldsymbol{\mu}_j\}_{j=1}^m \in \mathbb{R}^p$, the K-means clustering algorithm iteratively updates assignments and cluster centroids as follows [24]:

(1) Update assignments: for $i = 1, \ldots, n$

$$\mathbf{x}_i \in \mathcal{S}_j \Leftrightarrow j \in \underset{j' \in \{1,\ldots,m\}}{\arg\min} \|\mathbf{x}_i - \boldsymbol{\mu}_{j'}\|_2$$

(2) Update cluster centroids: for $j = 1, \ldots, m$

$$\boldsymbol{\mu}_j = \frac{1}{|\mathcal{S}_j|} \sum_{\mathbf{x}_i \in \mathcal{S}_j} \mathbf{x}_i \tag{5.12}$$

where $|\mathcal{S}_j|$ denotes the number of data points in the cluster $\mathcal{S}_j$ and $\boldsymbol{\mu}_j$ is the sample mean of the j-th cluster.

For large-scale data sets with large p and/or n, the memory requirements and computation cost of performing the K-means clustering algorithm become expensive [8, 123, 55]. First, the K-means algorithm requires several passes on the entire data set and thus the data set should often be stored in a centralized location which takes $\mathcal{O}(pn)$ memory. Second, the time complexity of K-means clustering is $\mathcal{O}(pnm)$ per iteration to partition the set of n data points into m clusters [131]. Hence, the high dimensionality of massive data sets provides considerable challenge to the design of memory and computation efficient alternatives for the Clustered Nyström method.

One promising strategy to address these obstacles is to use random projections of the data for constructing a small set of new features [1, 113, 151, 109]. In this case, for some parameter $p' < p$, the data matrix $\mathbf{X}$ is multiplied on the left by a random zero-mean matrix $\mathbf{H} \in \mathbb{R}^{p' \times p}$ in order to compute a low-dimensional representation:

$$\widehat{\mathbf{X}} = \mathbf{H}\mathbf{X} = [\mathbf{H}\mathbf{x}_1, \ldots, \mathbf{H}\mathbf{x}_n] \in \mathbb{R}^{p' \times n}.$$

The columns of $\widehat{\mathbf{X}} = [\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_n]$ are known as sketches or compressive measurements [42] and the random map $\mathbf{H}$ preserves the geometry of data under certain conditions [132]. The task of clustering is then performed on these low-dimensional data points by minimizing $E(\widehat{\mathbf{X}}, \mathcal{S}) = \sum_{i=1}^{n} \|\widehat{\mathbf{x}}_i - \mu(\widehat{\mathbf{x}}_i)\|_2^2$, which partitions the data points in the reduced space into m clusters. After finding the partition in the reduced space, the same partition is used on the original data points and the cluster centroids in the original space are calculated using Equation 5.12 at computational cost $\mathcal{O}(np)$.

In this work, we introduce a random-projection-type Clustered Nyström method, called "Randomized Clustered Nyström," for generating landmark points. In the first step of our method, a random sign matrix $\mathbf{H} \in \mathbb{R}^{p' \times p}$ whose entries are independent realizations of

$\{\pm 1/\sqrt{p'}\}$ Bernoulli random variables is constructed:

$$H_{ij} = \begin{cases} +1/\sqrt{p'} & \text{with probability } 1/2, \\ -1/\sqrt{p'} & \text{with probability } 1/2. \end{cases} \tag{5.13}$$

Next, the product $\mathbf{HX}$ is computed to find the low-dimensional sketches $\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_n \in \mathbb{R}^{p'}$. The standard implementation of matrix multiplication costs $\mathcal{O}(p'pn)$. The matrix multiplication can also be performed in parallel which leads to noticeable accelerations in practice [68]. Moreover, it is possible to use the mailman algorithm [88] which takes advantage of the binary-nature of $\mathbf{H}$ to further speed up the matrix multiplication. In our experiments, we use Intel MKL BLAS version 11.2.3 which is bundled with MATLAB, which we found to be sufficiently optimized and does not form a bottleneck in the computational cost.

In the second step, the K-means clustering algorithm is performed on the projected low-dimensional data $\widehat{\mathbf{X}} = [\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_n]$ to partition the data set:

$$\widehat{\mathcal{S}}^{opt} \approx \arg\min_{\mathcal{S}} E(\widehat{\mathbf{X}}, \mathcal{S}),$$

where $\widehat{\mathcal{S}}^{opt} = \{\widehat{\mathcal{S}}_1^{opt}, \ldots, \widehat{\mathcal{S}}_m^{opt}\}$ is the resulting m-partition. We cannot guarantee that K-means returns the globally optimal partition as the problem is NP-hard [40] but seeding using K-means++ [14] guarantees a partition with expected objective within a $\log(m)$ factor of the optimal one, and other variants of K-means, under mild assumptions [104], can either efficiently guarantee a solution within a constant factor of optimal, or guarantee solutions arbitrarily close to optimal, so-called polynomial-time approximation schemes (PTAS). Lastly, the landmark points are generated by computing the sample mean of data points:

$$\mathbf{z}_j = \left(1/|\widehat{\mathcal{S}}_j^{opt}|\right) \sum_{\mathbf{x}_i \in \widehat{\mathcal{S}}_j^{opt}} \mathbf{x}_i, \quad j = 1, \ldots, m. \tag{5.14}$$

The proposed "Randomized Clustered Nyström" method is summarized in Algorithm 5.3. In our method, the "compression factor" γ is defined as the ratio of parameter p' to the ambient dimension p, i.e., $\gamma := p'/p < 1$. Regarding the memory complexity, our method

Algorithm 5.3 Randomized Clustered Nyström

Input: data set $\mathbf{X}$, number of landmark points m, compression factor $\gamma < 1$
Output: landmark points $\mathbf{Z}$

1: Set $p' = \gamma p$ (round to nearest integer)
2: Generate a random sign matrix $\mathbf{H} \in \mathbb{R}^{p' \times p}$ as in Equation 5.13
3: Compute $\widehat{\mathbf{X}} = \mathbf{H}\mathbf{X} \in \mathbb{R}^{p' \times n}$
4: Perform K-means clustering on $\widehat{\mathbf{X}} = [\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_n]$ to get $\widehat{\mathcal{S}}^{opt}$
5: Compute the sample mean in the original space as in Equation 5.14
6: $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_m] \in \mathbb{R}^{p \times m}$

requires only two passes on the data set $\mathbf{X}$, the first to compute the low-dimensional sketches (step 3), and the second for the sample mean (step 5). In fact, our Randomized Clustered Nyström only stores the low-dimensional sketches which takes $\mathcal{O}(p'n)$ space, whereas the Clustered Nyström method has memory complexity of $\mathcal{O}(pn)$, meaning our method reduces the memory complexity by a factor of $1/\gamma$. In terms of time complexity, the computation cost of K-means on the dimension-reduced data in our method is $\mathcal{O}(p'nm)$ per iteration compared to the cost $\mathcal{O}(pnm)$ in the Clustered Nyström method, so the speedup is up to $1/\gamma$ (the exact amount depends on the number of iterations, since we must amortize the cost of the one-time matrix multiply $\mathbf{HX}$).

Thus, our proposed method for generating landmark points provides a tunable parameter γ to reduce the memory and computation cost of the Clustered Nyström method. Next, we study and characterize the "tradeoffs" between accuracy of low-rank approximations and the memory/computation savings in our proposed method. In particular, the following theorem presents an error bound on the Nyström low-rank approximation for a set of landmark points generated via our Randomized Clustered Nyström method (Algorithm 5.3).

Theorem 5.1 (Randomized Clustered Nyström Method)**.** *Assume that the kernel function κ satisfies Equation 5.7. Consider the data set $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ and the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ with entries $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The optimal partitioning of $\mathbf{X}$ into m clusters is*

denoted by $\mathcal{S}^{opt}$*:*

$$\mathcal{S}^{opt} = \arg\min_{\mathcal{S}} E\left(\mathbf{X}, \mathcal{S}\right), \text{ where } E\left(\mathbf{X}, \mathcal{S}\right) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mu(\mathbf{x}_i)\|_2^2. \tag{5.15}$$

Let us generate a random sign matrix $\mathbf{H} \in \mathbb{R}^{p' \times p}$ *as in Equation 5.13 with* $p' = \mathcal{O}(m/\varepsilon^2)$ *for some parameter* $\varepsilon \in (0, 1/3)$*. The Randomized Clustered Nyström method computes the product* $\widehat{\mathbf{X}} = \mathbf{H}\mathbf{X}$ *to generate a set of* m *landmark points* $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_m]$ *by partitioning of* $\widehat{\mathbf{X}} \in \mathbb{R}^{p' \times n}$ *into* m *clusters. We assume that the partitioning* $\widehat{\mathcal{S}}^{opt}$ *of* $\widehat{\mathbf{X}}$ *leads to* $E(\widehat{\mathbf{X}}, \widehat{\mathcal{S}}^{opt})$ *within a constant factor of the optimal value, cf. [14, 104]. Given matrices* $\mathbf{C} \in \mathbb{R}^{n \times m}$ *and* $\mathbf{W} \in \mathbb{R}^{m \times m}$ *whose entries are* $C_{ij} = \kappa(\mathbf{x}_i, \mathbf{z}_j)$ *and* $W_{ij} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$*, the Nyström approximation error is bounded with probability at least* 0.96 *over the randomness of* $\mathbf{H}$*:*

$$\mathcal{E} \stackrel{def}{=} \|\mathbf{K} - \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T\|_F \leq \eta_1 \sqrt{(2+\varepsilon) E\left(\mathbf{X}, \mathcal{S}^{opt}\right)} + \eta_2 (2+\varepsilon) E\left(\mathbf{X}, \mathcal{S}^{opt}\right), \tag{5.16}$$

where η_1 *and* η_2 *are two positive constants.*

Proof. Based on Proposition 5.1, we get the following approximation error for the Randomized Clustered Nyström method:

$$\mathcal{E} \leq \eta_1 \sqrt{E(\mathbf{X}, \widehat{\mathcal{S}}^{opt})} + \eta_2 E(\mathbf{X}, \widehat{\mathcal{S}}^{opt}), \tag{5.17}$$

where $\widehat{\mathcal{S}}^{opt}$ is the optimal partitioning of the reduced data $\widehat{\mathbf{X}}$ and $E(\mathbf{X}, \widehat{\mathcal{S}}^{opt})$ represents the total quantization error when $\widehat{\mathcal{S}}^{opt}$ is used to cluster the high-dimensional data $\mathbf{X}$. We assume the partitioning in the reduced data set is within a constant factor of optimal, so this constant is absorbed into η_1 and η_2. In [26], it is shown that by choosing $p' = \mathcal{O}(m/\varepsilon^2)$ dimensions for the random projection matrix $\mathbf{H}$, the following inequality holds with probability at least 0.96 over the randomness of $\mathbf{H}$:

$$E(\mathbf{X}, \widehat{\mathcal{S}}^{opt}) \leq (2+\varepsilon) E(\mathbf{X}, \mathcal{S}^{opt}). \tag{5.18}$$

Thus, employing the above inequality in Equation 5.17 completes the proof. □

The error bound in Theorem 5.1 reveals important insights about the performance of our proposed method. Although our Randomized Clustered Nyström generates landmark points based on the random projections of data, we can relate the approximation error to the total quantization error of partitioning the original data points. In fact, our results show that the random projections of original data points into $\mathbb{R}^{p'}$ with $p' = \mathcal{O}(m)$ yields an approximation which is close to the one obtained by the Clustered Nyström method (Proposition 5.1). Interestingly, the dimension of the reduced data p' is independent of the ambient dimension p and depends only on m (the number of landmark points) and ε (the distortion factor). As a result, for high-dimensional data sets with large p, the dimension of reduced data p' can be fixed based on the desired number of landmark points and accuracy.

5.6 Numerical Experiments

In this section, we present experimental results comparing our Randomized Clustered Nyström with a few other sampling methods such as the Clustered Nyström method and uniform sampling. Our proposed approach is implemented in MATLAB with the C/mex implementation for computing the sample mean in step 5 of Algorithm 5.3. To perform the K-means clustering algorithm, we use MATLAB's built-in function `kmeans` and the maximum number of iterations is set to 10. This function utilizes the K-means++ algorithm [14] for cluster center initialization which improves the performance over random initializations. Note that the K-means++ algorithm needs m passes over the data to choose m initial cluster centers. Since our Randomized Clustered Nyström performs K-means clustering on the low-dimensional random projections (step 4 of Algorithm 5.3), the overall number of passes on the data set will remain two.

The performance of our proposed Randomized Clustered Nyström method is demonstrated on two different tasks: low-rank approximation of kernel matrices (Section 5.6.1) and kernel ridge regression (Section 5.6.2). All the experiments are conducted on a desktop computer with two Intel Xeon EF-2650 v3 CPUs at 2.4–3.2 GHz and 8 cores.

Table 5.1: Summary of data sets used in Section 5.6.1.

data set	p	n
`dna`	180	2,000
`protein`	357	17,766
`mnist`	784	60,000
`epsilon`	2,000	60,000

5.6.1 Kernel approximation quality

In this section, we study the accuracy and efficiency of our Randomized Clustered Nyström on the low-rank approximation of kernel matrices in the form of $\mathbf{K} \approx \mathbf{L}\mathbf{L}^T$, where $\mathbf{L} \in \mathbb{R}^{n \times r}$ for target rank r. Experiments are conducted on four data sets from the LIBSVM archive [29], listed in Table 5.1. In all experiments, similar to [147], the Gaussian kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/c)$ is used with the parameter c chosen as the averaged squared distances between all the data points and sample mean. The approximation accuracy is measured by the normalized kernel approximation error in terms of the Frobenius norm: $\|\mathbf{K} - \mathbf{L}\mathbf{L}^T\|_F / \|\mathbf{K}\|_F$. We report the mean and standard deviation of approximation error over 50 trials because all sampling methods in the Nyström method involve some randomness.

For two data sets `dna` and `protein` with the total number of data points less than 20,000, the entire kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ can be stored and manipulated in the main memory of the computer. Thus, the accuracy of our proposed method for various values of the compression factor γ is compared with:

(1) The SVD, where the best rank-r approximation of the kernel matrix $\mathbf{K}$ is obtained via the exact eigenvalue decomposition or SVD;

(2) Uniform sampling [143], where m landmark points are selected uniformly at random without replacement from n data points;

(3) Column-norm sampling [46], where m columns of the kernel matrix $\mathbf{K}$ are sampled

with weights proportional to ℓ_2 norm of columns;

(4) Clustered Nyström [147], where m landmark points are generated using centroids resulting from K-means clustering on the original data set.

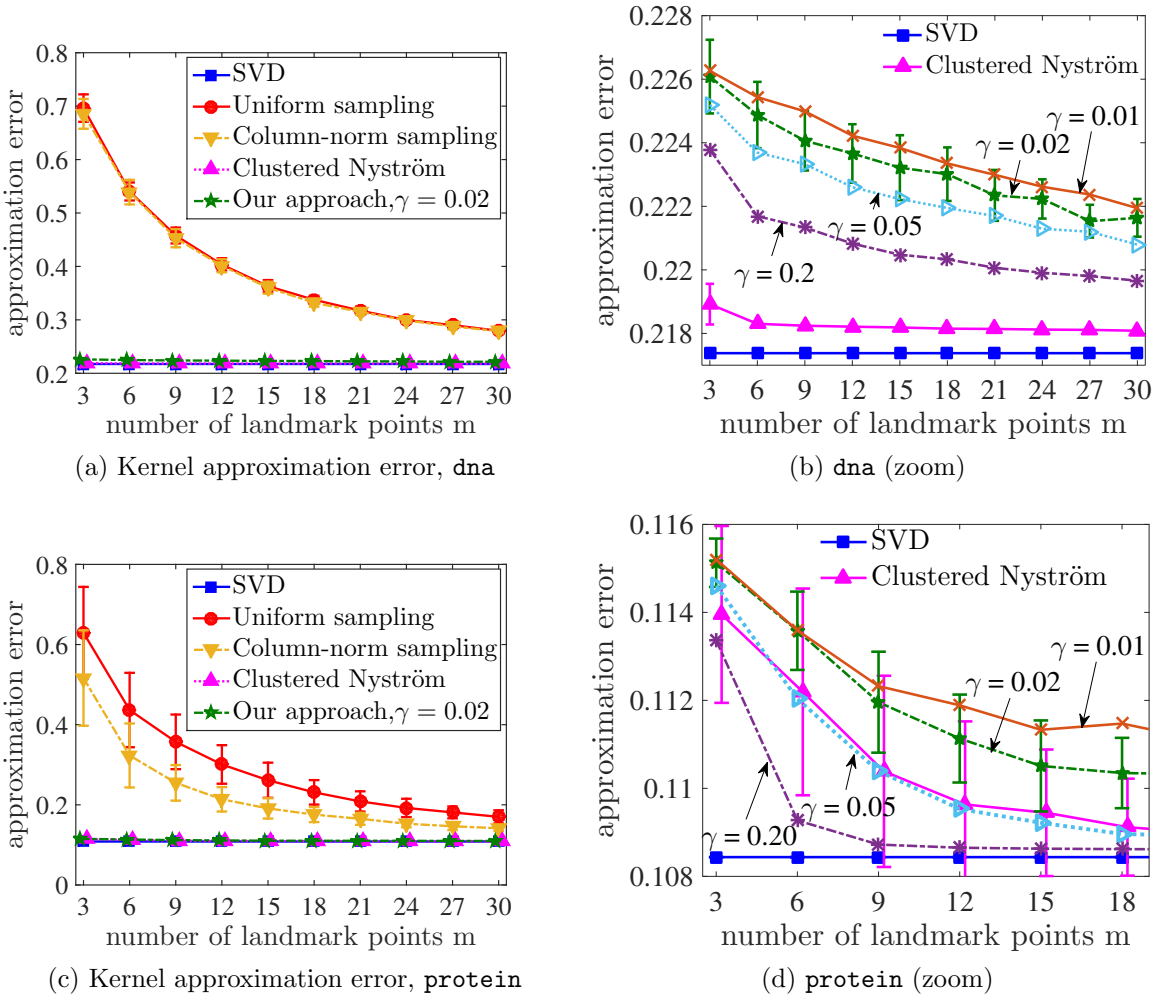
Also, for all choices of landmark points (Randomized Clustered Nyström and options 2, 3, 4 above) with m greater than the target rank r, our proposed "Nyström via QR decomposition" (Algorithm 5.2) is used to restrict the resulting Nyström approximation to have rank at most r, cf. Section 5.4.

For the two large-scale data sets `mnist` and `epsilon`, it takes approximately 29 GB space to store the kernel matrix. Thus, the storage and manipulation of $\mathbf{K}$ in the main memory becomes too costly and our proposed method is compared with just the two sampling methods which do not need access to the entire kernel matrix, namely uniform sampling and Clustered Nyström.

5.6.1.1 Data sets: `dna` and `protein`

The first example demonstrates the effectiveness of various sampling methods on improving the accuracy of the Nyström method by increasing the number of landmark points. The mean and standard deviation of kernel approximation error are reported in Figure 5.3 for varying number of landmark points m with fixed target rank $r = 3$. The results in Figure 5.3a and Figure 5.3c show that both Clustered Nyström and our proposed method with $\gamma = 0.02$ ($p' = 4$ for `dna` and $p' = 7$ for `protein`) improve the accuracy of the Nyström method over uniform sampling and column-norm sampling. In fact, the accuracy of our proposed method and Clustered Nyström reaches the accuracy of the best rank-r approximation (SVD) for small values of m, e.g., $m = r$. The uniform sampling method does not reach this accuracy even if it uses a large number of landmark points such as $m = 10r = 30$.

To further investigate the tradeoffs between accuracy and efficiency of our proposed method, the mean and standard deviation of kernel approximation error for a few values



(a) Kernel approximation error, `dna`

(b) `dna` (zoom)

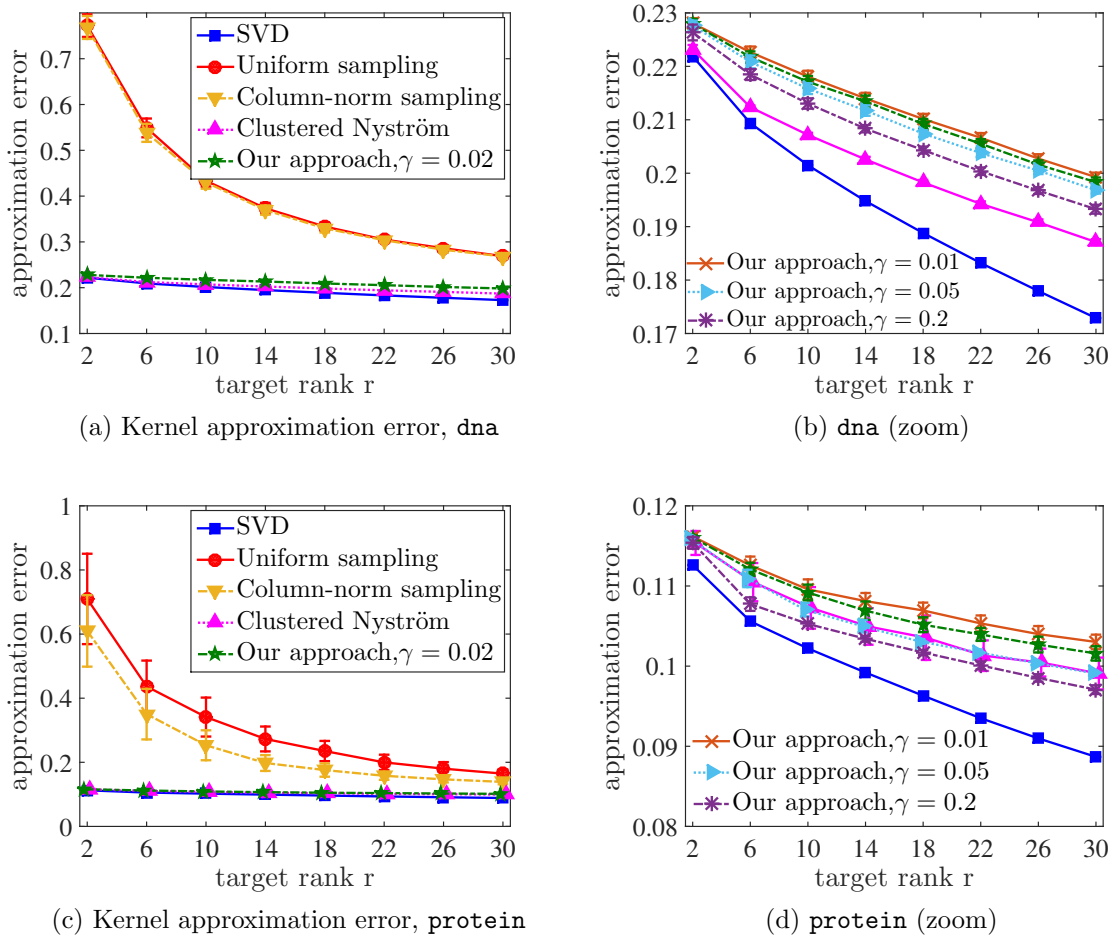
(c) Kernel approximation error, `protein`

(d) `protein` (zoom)

Figure 5.3: Kernel approximation error for varying number of landmark points m with target rank $r = 3$ on `dna` and `protein` data sets.

of the compression factor γ from 0.01 to 0.2 are presented in Figure 5.3b and Figure 5.3d (error bars for $\gamma = 0.01, 0.05, 0.2$ have been omitted for clarity). As the compression factor γ (equivalently, p') increases, the approximation error decreases which is consistent with our theoretical results in Theorem 5.1. However, small values of γ in our method, such as $\gamma = 0.01$, lead to accurate low-rank approximations with savings in memory and computation by a factor of $1/\gamma = 100$. As a final note, it is observed that our method with $\gamma = 0.2$ performs slightly better than the Clustered Nyström method on the `protein` data set. This is mainly due to the fact that the performance of K-means clustering depends on the starting

points. It is possible for K-means to reach a local minimum solution, where a better solution with the lower value of objective function exists. In practice, one can increase the number of random initializations and select the clustering with the lowest value of objective function. The difference in performance between Clustered Nyström and our method with $\gamma = 0.2$ disappears if we instead take the best result out of 20 independent initializations.



(a) Kernel approximation error, `dna`

(b) `dna` (zoom)

(c) Kernel approximation error, `protein`

(d) `protein` (zoom)

Figure 5.4: Kernel approximation error for various values of target rank r and fixed $m = r$ on the `dna` and `protein` data sets.

The second example, shown in Figure 5.4, demonstrates the performance of our proposed method for various values of target rank r from 2 to 30 and fixed $m = r$. Based on Figure 5.4a and Figure 5.4c, it is clear that both our method and Clustered Nyström

method provide improved approximation accuracy over uniform sampling and column-norm sampling techniques. In fact, we see that both our method and Clustered Nyström method have roughly the same accuracy as the best rank-r approximation (SVD) for all values of the target rank.

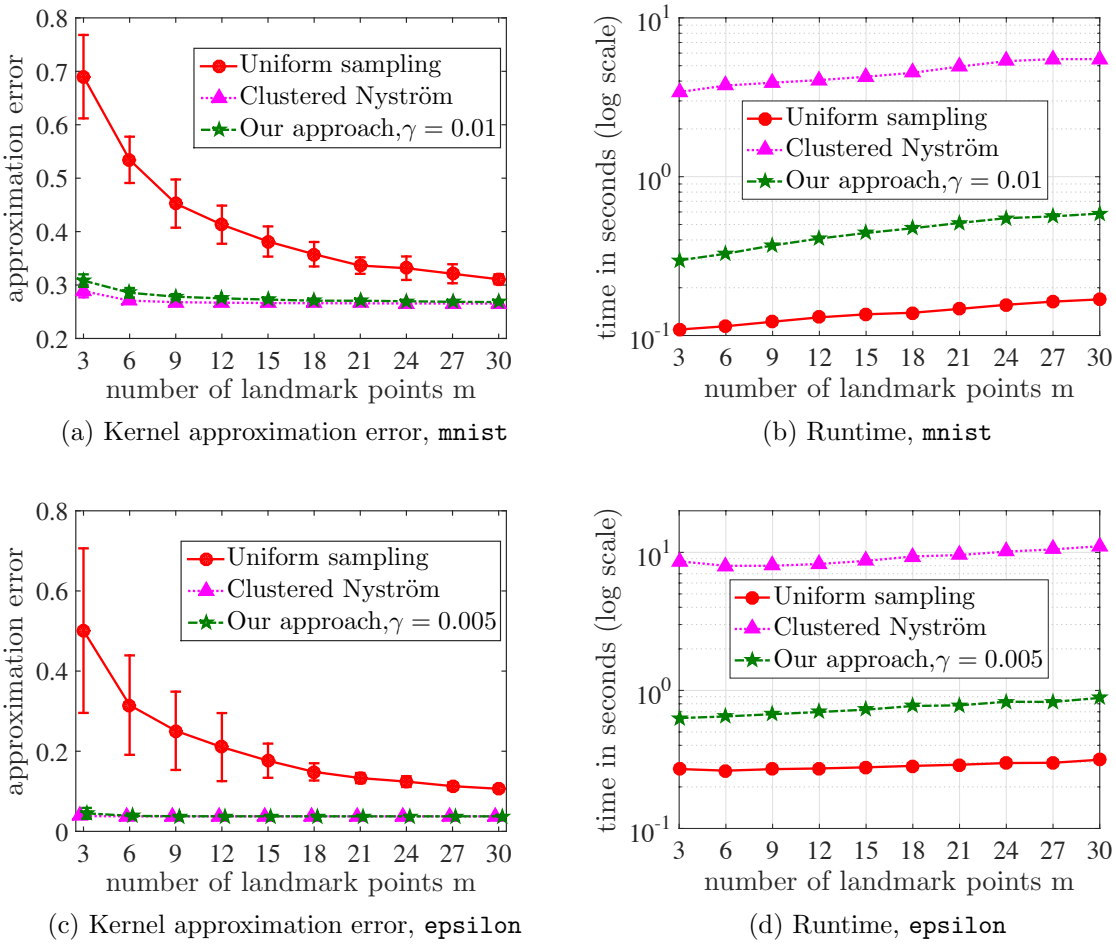
We also report the mean and standard deviation of kernel approximation error in Figure 5.4b and Figure 5.4d for varying values of compression factor γ from 0.01 to 0.2. As the compression factor γ (or the number of dimensions p') increases, the kernel approximation error decreases as prescribed by our theoretical results in Theorem 5.1. Also, we see that small values of compression factor γ result in accurate low-rank approximations which lead to memory and computation savings by a factor of $1/\gamma$ in comparison with the Clustered Nyström method.

5.6.1.2 Data sets: `mnist` and `epsilon`

The accuracy and time complexity of our Randomized Clustered Nyström method are demonstrated on two large-scale examples. The parameter γ is set to 0.01 for the `mnist` data set ($p' = 8$) and 0.005 for the `epsilon` data set ($p' = 10$).

In the first example, the normalized kernel approximation error and computation time are reported in Figure 5.5 for various values of m and fixed target rank $r = 3$. In Figure 5.5a and Figure 5.5c, we observe that our proposed method outperforms uniform sampling and has almost the same accuracy as the Clustered Nyström method for all values of m. However, the runtime of our proposed method is reduced by an order of magnitude compared to the Clustered Nyström method (Figure 5.5b and Figure 5.5d). Thus, our proposed method provides significant memory and computation savings with little loss in accuracy compared to the Clustered Nyström method. While our randomized method spends more time than uniform sampling to find a small set of informative landmark points, it provides improved approximation accuracy. Thus, these empirical results suggest a tradeoff between time and space requirements of our proposed method and uniform sampling. For example, our method

with $m = r$ landmark points outperforms uniform sampling with $m = 10r$ on the `epsilon` data set.



(a) Kernel approximation error, `mnist`

(b) Runtime, `mnist`

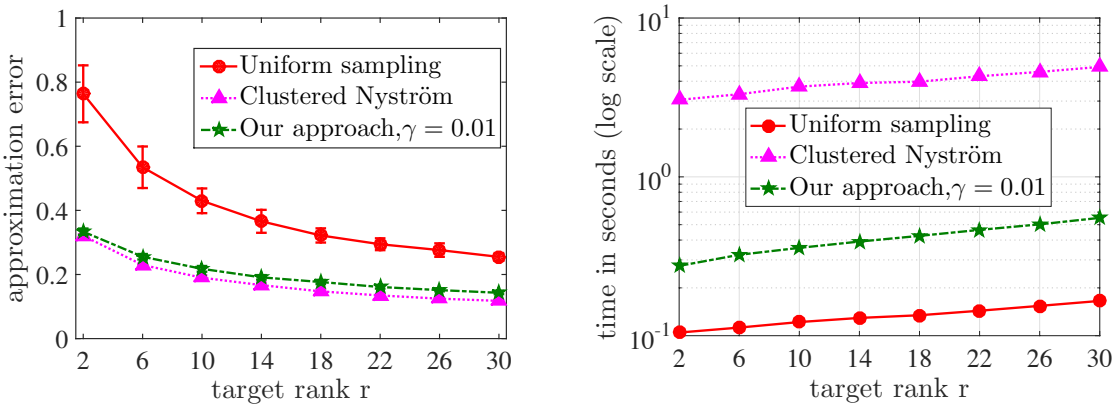
(c) Kernel approximation error, `epsilon`

(d) Runtime, `epsilon`

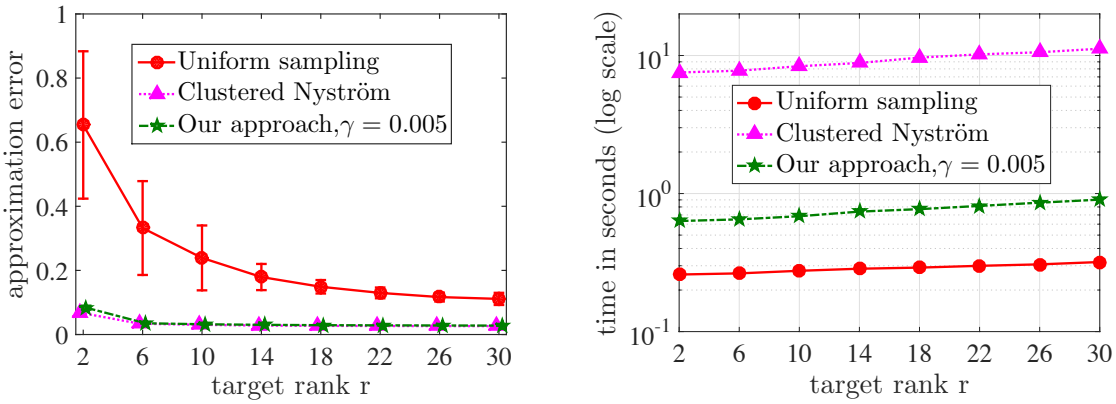
Figure 5.5: Kernel approximation error and runtime for varying number of landmark points m with target rank $r = 3$ on `mnist` and `epsilon` data sets.

In the second example, our proposed method is compared with uniform sampling and Clustered Nyström for various values of the target rank r from 2 to 30 (fixing $m = r$) and results are reported in Figure 5.6. As we see in Figure 5.6a and Figure 5.6c, our proposed method outperforms uniform sampling for all values of the target rank and has almost the same accuracy as the Clustered Nyström method. Similar to the previous example, the time complexity of our proposed method is decreased by an order of magnitude compared

to the Clustered Nyström. Hence, our Randomized Clustered Nyström provides improved approximation accuracy, while being more efficient in terms of memory and computation than the Clustered Nyström method.



(a) Kernel approximation error, `mnist` (b) Runtime, `mnist`



(c) Kernel approximation error, `epsilon` (d) Runtime, `epsilon`

Figure 5.6: Kernel approximation error and runtime for various values of target rank r and fixed $m = r$ on `mnist` and `epsilon` data sets.

5.6.2 Kernel ridge regression

In this section, we present experimental results on the performance of various sampling methods when used with kernel ridge regression. In the supervised learning setting, a set of instance-label pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ are given, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. Kernel ridge regression

proceeds by generating $\boldsymbol{\alpha}^*$ which solves the dual optimization problem [119]:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \ \boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha}^T\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T\mathbf{y}, \tag{5.19}$$

where $\mathbf{K}$ is the kernel matrix, $\mathbf{y} = [y_1, \ldots, y_n]^T \in \mathbb{R}^n$ is the response vector, and $\lambda > 0$ is the regularization parameter. The problem admits the closed-form solution $\boldsymbol{\alpha}^* = (\mathbf{K} + \lambda\mathbf{I}_{n\times n})^{-1}\mathbf{y}$, which requires the computation of $\mathbf{K} \in \mathbb{R}^{n\times n}$ and the $n \times n$ system to solve. Memory and computation cost can be reduced by using the low-rank approximation of the kernel matrix $\mathbf{K} \approx \mathbf{L}\mathbf{L}^T$, where $\mathbf{L} \in \mathbb{R}^{n\times r}$, to generate an approximate solution (cf. Equation 5.2):

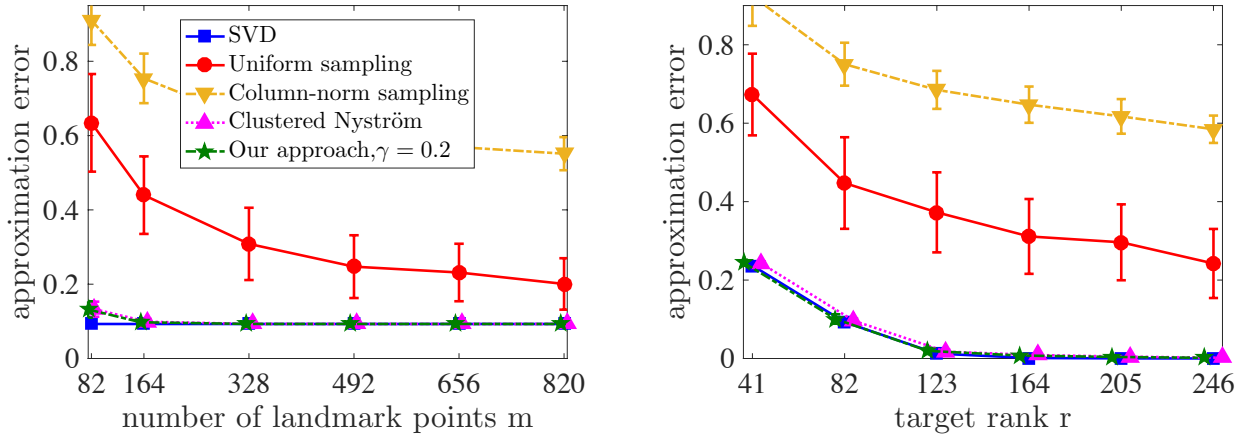
$$\widehat{\boldsymbol{\alpha}} = \lambda^{-1}\left(\mathbf{I}_{n\times n} - \mathbf{L}\big(\mathbf{L}^T\mathbf{L} + \lambda\mathbf{I}_{r\times r}\big)^{-1}\mathbf{L}^T\right). \tag{5.20}$$

The authors in [37] analyzed the effect of such low-rank approximations on the accuracy of the approximate solution $\widehat{\boldsymbol{\alpha}}$.

Here, we empirically compare the accuracy of our Randomized Clustered Nyström with a few other sampling methods. The approximation error is defined as $\|\widehat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*\|_2/\|\boldsymbol{\alpha}^*\|_2$ and we report the mean and standard deviation of the approximation error over 50 trials. Two data sets from the LIBSVM archive [29] are considered for regression: (1) `cpusmall` and (2) `E2006-tfidf`. The former data set consists of $n = 8{,}192$ samples with $p = 12$ and we increase the dimensionality to $p = 48$ by repeating each entry 4 times. The `E2006-tfidf` data set contains $n = 5{,}363$ samples with $p = 150{,}360$. As before, the Gaussian kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/c)$ is used with the parameter c chosen as the averaged squared distances as in the previous section. The regularization parameter λ is set to $1/4$.

In Figure 5.7, the approximation error on the `cpusmall` data set is reported when $\gamma = 0.2$ in our method ($p' = 10$) for two cases: (1) fixed target rank $r/n = 0.01$ and varying number of landmark points m (Figure 5.7a); (2) various values of the target rank r from $0.005n$ to $0.03n$ and fixed $m = 2r$ (Figure 5.7b). The performance of our Randomized Clustered Nyström is significantly better than that of the uniform sampling and column-norm

sampling approaches. In fact, our proposed method is as nearly accurate as the best rank-r approximation (SVD) for just $m = 2r$. There is no significant difference in performance between our method and the full Clustered Nyström method.



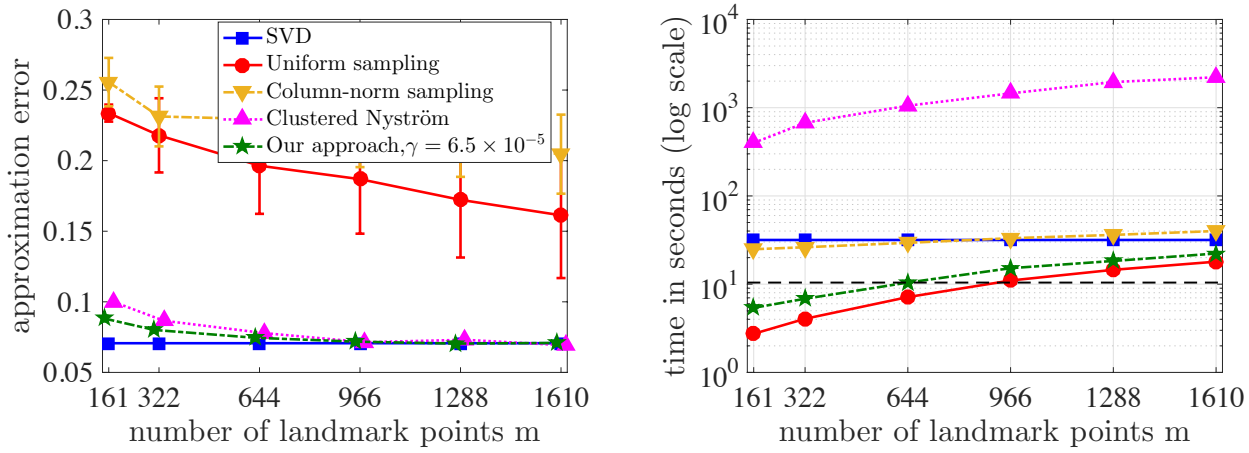
(a) Approximation error of $\boldsymbol{\alpha}^*$, $r = 82$

(b) Approximation error of $\boldsymbol{\alpha}^*$, $m = 2r$

Figure 5.7: Kernel ridge regression on the `cpusmall` data set.

In Figure 5.8, the approximation error on the `E2006-tfidf` data set is presented for the target rank $r/n = 0.03$ and varying number of landmark points. The parameter γ in our Randomized Clustered Nyström is set to 6.5×10^{-5} which means that $p' = 10$. Our method is much more accurate than both uniform and column-norm sampling as shown in Figure 5.8a. Moreover, based on Figure 5.8b, our method reduces the computational complexity of the Clustered Nyström method by two orders of magnitude since Clustered Nyström performs K-means on a very high-dimensional data set.

We draw a dashed line in Figure 5.8b to find the values of m for which our method and uniform sampling have the same running time. We see that $m = 644$ in our method, which leads to mean error 0.074 and standard deviation 0.001, has the same running time as $m = 966$ in the uniform sampling, with mean 0.187 and standard deviation 0.038. This is an example of our method performing both more accurately and more efficiently than the alternatives.



(a) Approximation error of $\boldsymbol{\alpha}^*$, $r = 161$

(b) Runtime, $r = 161$

Figure 5.8: Kernel ridge regression on the `E2006-tfidf` data set.

Chapter 6

Conclusion

In this section, the main contributions in each chapter are summarized. Furthermore, an efficient method for estimating active subspaces [33] is proposed using only random observations of gradient vectors. The introduced method is based on the bi-linear representation of low-rank gradient matrices with an initialization step motivated by the results in Chapter 4 for alternating minimization. As future work, we pursue both theoretical analysis and additional numerical experiments.

Chapter 2: In this chapter, we presented a memory- and computation-efficient approach for estimation of covariance matrices and principal components via very sparse random projections. This approach simultaneously reduces substantially the required memory and computation for PC estimation, while still providing high accuracy. More importantly, it allows us to rigorously analyze each of memory, computation, and accuracy in terms of the sparsity of the projection. Thus, we have been able to give provable tradeoffs between memory, computation, and accuracy. Furthermore, a user of this approach could even use the sparsity of the projections to tune to any desired point on this three-way tradeoff.

Chapter 3: In this chapter, a novel theoretical framework for compressive dictionary learning problem is presented. These results give us valuable insights about the effect of various parameters on the performance of dictionary learning algorithms. Also, we presented a memory- and computation-efficient dictionary learning algorithm for modern data settings via very sparse random projections. Our simulation results show that this approach holds

much potential for analyzing massive amounts of high-dimensional data.

Chapter 4: In this chapter, 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.

Chapter 5: In this chapter, we presented two complementary methods to improve the quality of Nyström low-rank approximations. The first method, "Nyström via QR Decomposition," finds the best rank-r approximation when the number of landmark points is greater than the target rank in the Nyström method. The experimental examples demonstrated the superior performance of our proposed method compared to the standard Nyström method. Also, for a fixed accuracy, the introduced method requires fewer landmark points which is of great importance for the efficiency of the Nyström method.

The second proposed method, "Randomized Clustered Nyström," is a randomized algorithm for generating landmark points, where the memory and computational complexity can be adjusted by using the compression factor. Based on our experiments, random projection of the input data points onto a low-dimensional space with 10 or smaller dimensions yields very accurate low-rank approximations. In fact, the accuracy of our proposed method is very close to the best rank-r approximation obtained by the exact eigenvalue decomposition or SVD of kernel matrices.

In the next section, a randomized algorithm for estimating active subspaces is introduced.

6.1 Estimating Active Subspaces with Randomized Gradient Sampling

In modern computer simulations, scientists and engineers seek to study the relationships between high-dimensional spaces of input parameters and quantities of interest. Due to the large number of input parameters and high cost of simulations, many methods have been proposed to reduce the dimension of the input parameter space. These methods often find small subsets or linear combinations of the input parameters that approximately preserve input-output relationships. This low-dimensional characterization of complex problems with hundreds or thousands of input parameters is a crucial tool for modern computer simulations.

6.1.1 Active subspaces

Active subspaces are powerful tools for identifying important directions in the high-dimensional space of input parameters [33, 34]. Let $\mathbf{x} \in \mathbb{R}^m$ be a vector of simulation inputs and assume that $f(\mathbf{x}) : \mathbb{R}^m \mapsto \mathbb{R}$ is the mapping between $\mathbf{x}$ and a quantity of interest. The active subspace is defined by the top $n < m$ eigenvectors of the following $m \times m$ symmetric

positive semidefinite matrix

$$\mathbf{C} = \int \nabla f(\mathbf{x}) \nabla f(\mathbf{x})^T \rho(\mathbf{x}) d\mathbf{x}, \tag{6.1}$$

where $\nabla f(\mathbf{x}) \in \mathbb{R}^m$ is the gradient vector and ρ is a user-specified probability density function. Consider the eigenvalue decomposition of $\mathbf{C} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^T$, where $\mathbf{\Lambda} \in \mathbb{R}^{m \times m}$ is the diagonal matrix of eigenvalues, listed in decreasing order, and $\mathbf{W} \in \mathbb{R}^{m \times m}$ contains m orthonormal eigenvectors. The matrix $\mathbf{W}$ can then be partitioned: $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2]$. The column space of $\mathbf{W}_1 \in \mathbb{R}^{m \times n}$ is the n-dimensional active subspace, where n is usually chosen so the first n eigenvalues are much larger than the remaining $m - n$ eigenvalues.

In high-dimensional settings, computing the integral in (6.1) for constructing the matrix $\mathbf{C}$ is impractical. Moreover, in some applications, the gradient vector $\nabla f(\mathbf{x})$ may not have a closed-form expression. In such cases, gradients can be approximated by the first-order finite difference with $m + 1$ function evaluations

$$\mathbf{e}_j^T \nabla f(\mathbf{x}) \approx \left(f(\mathbf{x} + h\mathbf{e}_j) - f(\mathbf{x})\right)/h, \quad j = 1, \ldots, m, \tag{6.2}$$

where $\mathbf{e}_j$ is the j-th canonical basis vector in $\mathbb{R}^m$ and $h > 0$ is the finite difference parameter.

In [36], it is shown that the matrix $\mathbf{C}$ and its eigenpairs can be approximated using the following Monte Carlo method. First, M samples $\mathbf{x}_1, \ldots, \mathbf{x}_M \in \mathbb{R}^m$ are drawn i.i.d. according to ρ. Then, $\nabla f_i := \nabla f(\mathbf{x}_i)$ are estimated via (6.2) using $M(m + 1)$ function evaluations to form

$$\widehat{\mathbf{C}} = \frac{1}{M} \sum_{i=1}^{M} \nabla f_i \nabla f_i^T = \widehat{\mathbf{W}} \widehat{\mathbf{\Lambda}} \widehat{\mathbf{W}}^T. \tag{6.3}$$

The leading n eigenvectors of $\widehat{\mathbf{C}}$ provide an accurate estimate of $\mathbf{W}_1$ when M is sufficiently large [36]. These eigenvectors are equivalent to computing the top left singular vectors of the gradient matrix

$$\mathbf{G} := [\nabla f_1, \ldots, \nabla f_M] \in \mathbb{R}^{m \times M}. \tag{6.4}$$

In this work, we show that the *low-rank* structure of the gradient matrix $\mathbf{G}$ allows us to find accurate estimates of active subspace using fewer function evaluations. In particular, we consider a scheme where only k entries of each gradient vector $\nabla f_i \in \mathbb{R}^m$ are computed uniformly at random, thus the total number of function evaluations required would be $M(k+1)$. To estimate the active subspace, $\mathbf{G}$ is written in a bi-linear form $\mathbf{G} = \mathbf{AB}$ and then alternating minimization [74] is used to find $\mathbf{A}$ and $\mathbf{B}$ that best fit the observed entries of gradient matrix. To further improve the performance for small values of k, we use an unbiased estimate of the left singular vectors of $\mathbf{G}$ as the initial point for alternating minimization.

6.1.2 Estimating active subspaces with gradient sampling

Let us define the linear measurement operator $\mathcal{L}(\cdot)$ as

$$\mathcal{L}(\mathbf{G}) := \left[\mathbf{R}_1^T \nabla f_1, \ldots, \mathbf{R}_M^T \nabla f_M\right] \in \mathbb{R}^{k \times M}, \tag{6.5}$$

where each sampling matrix $\mathbf{R}_i \in \mathbb{R}^{m \times k}$ contains k canonical basis vectors in $\mathbb{R}^m$ chosen uniformly at random without replacement. Given the incomplete observations and the bi-linear parameterization of $\mathbf{G}$, our goal is to minimize

$$\min_{\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{n \times M}} \|\mathcal{L}(\mathbf{G}) - \mathcal{L}(\mathbf{AB})\|_F. \tag{6.6}$$

This problem can be reformulated by using the vector operator and kronecker product

$$\begin{aligned} \|\mathcal{L}(\mathbf{G}) - \mathcal{L}(\mathbf{AB})\|_F &= \|\mathbf{y} - \mathbf{R}(\mathbf{I}_{M \times M} \otimes \mathbf{A})\mathrm{vec}(\mathbf{B})\|_2 \\ &= \|\mathbf{y} - \mathbf{R}(\mathbf{B}^T \otimes \mathbf{I}_{m \times m})\mathrm{vec}(\mathbf{A})\|_2, \end{aligned} \tag{6.7}$$

where $\mathbf{y} = \mathbf{R}\mathrm{vec}(\mathbf{G})$ and $\mathbf{R} \in \mathbb{R}^{kM \times mM}$ contains all sampling matrices $\mathbf{R}_i^T$, $i = 1, \ldots, M$, on its main diagonal. Thus, our method iteratively keep one of $\mathbf{A}, \mathbf{B}$ fixed and optimize over the other. Each subproblem is convex and can be solved efficiently

$$\begin{aligned} \mathrm{vec}(\mathbf{B}) &\leftarrow \left[\mathbf{R}(\mathbf{I}_{M \times M} \otimes \mathbf{A})\right]^\dagger \mathbf{y}, \\ \mathrm{vec}(\mathbf{A}) &\leftarrow \left[\mathbf{R}(\mathbf{B}^T \otimes \mathbf{I}_{m \times m})\right]^\dagger \mathbf{y}. \end{aligned} \tag{6.8}$$

In [35], estimation of active subspaces is considered in a similar framework where each $\mathbf{R}_i \in \mathbb{R}^{m\times k}$ has independent standard Gaussian entries. As we see from the update rule in (6.8), our method is more efficient in terms of computation and memory. The proposed method in [35] must store Mmk nonzero entries of matrix $\mathbf{R}$, whereas our method requires only Mk nonzero entries to be stored. Similarly, our method reduces the cost of matrix-matrix multiplications. We expect our proposed sampling method to perform as well as Gaussian samples when the active subspaces are incoherent with the standard basis.

Another contribution of our work is the novel initialization step based on Chapter 4. The initial iterate for alternating minimization is preferred to be chosen based on a *good* estimate of $\mathbf{A}$, rather than random initializations, to guarantee the convergence [74]. In Chapter 4, an unbiased estimator for the matrix $\widehat{\mathbf{C}}$ is presented

$$\widetilde{\mathbf{\Sigma}} := \widetilde{\mathbf{C}} - \eta\,\mathrm{diag}(\widetilde{\mathbf{C}}),\quad \eta = \frac{m-k}{m-1}, \tag{6.9}$$

where $\mathrm{diag}(\widetilde{\mathbf{C}})$ represents the matrix formed by zeroing all but the diagonal elements of $\widetilde{\mathbf{C}}$, which is defined as

$$\widetilde{\mathbf{C}} := \frac{m(m-1)}{k(k-1)}\frac{1}{M}\sum_{i=1}^{M}(\mathbf{R}_i\mathbf{R}_i^T\nabla f_i)(\mathbf{R}_i\mathbf{R}_i^T\nabla f_i)^T. \tag{6.10}$$

6.1.3 Numerical experiments

Let $\mathbf{H} \in \mathbb{R}^{m\times m}$ be symmetric positive semidefinite and $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$, defined on the domain $\mathbf{x} \in [-1,1]^m$ with a uniform density ρ. Thus, the gradient is $\nabla f(\mathbf{x}) = \mathbf{H}\mathbf{x}$. The eigenvalues of $\mathbf{C}$ in are the eigenvalues of $\mathbf{H}$, squared and divided by 3. Moreover, the eigenvectors of $\mathbf{C}$ and $\mathbf{H}$ are identical. The matrix $\mathbf{H}$ is constructed so that its eigenvalues decay at a slow rate, except for a large gap between the fifth and sixth eigenvalues. We set parameters $m = 100$, $M = 2000$, and $n = 5$. The subspace estimation error is defined as $\mathcal{E} := \|\widehat{\mathbf{W}}_1\widehat{\mathbf{W}}_1^T - \widetilde{\mathbf{W}}_1\widetilde{\mathbf{W}}_1^T\|_2$, where $\widetilde{\mathbf{W}}_1$ is the active subspace estimate using incomplete gradients. In Fig. 6.1, the mean estimation error over 100 trials is reported for various values of measurements k. Alternating minimization with 20 iterations is used in two cases: (1)

our proposed initial point based on (6.9), and (2) random initialization based on a Gaussian matrix.

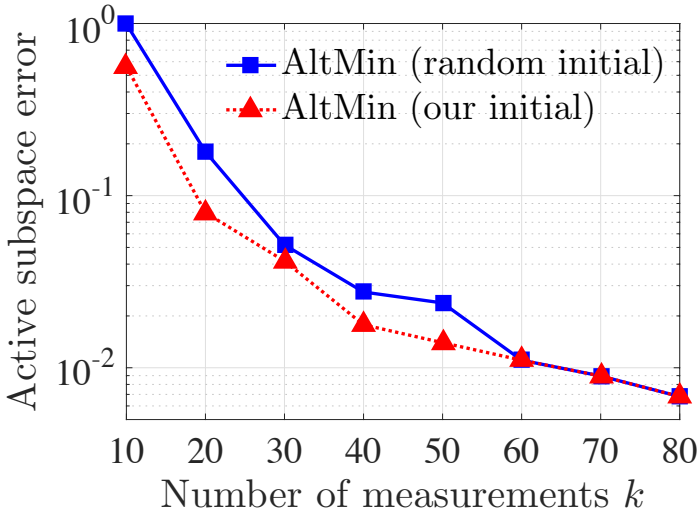


Figure 6.1: Active subspace estimation error for varying number of measurements k and fixed dimension $m = 100$.

A good initialization point becomes more important for small values of k, which are crucial for large-scale problems. For example, at $k = 20$, the mean estimation errors for our proposed initial point and random initialization are 0.08 and 0.18, respectively. Thus, our initialization procedure reduces the error by almost a factor of 2 in this case.

As future work, we pursue both theoretical analysis and comprehensive numerical experiments.

Bibliography

[1] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. Journal of Computer and System Sciences, 66(4):671–687, 2003. 16, 30, 75, 121, 143

[2] D. Achlioptas, Z. Karnin, and E. Liberty. Near-optimal entrywise sampling for data matrices. In Advances in Neural Information Processing Systems (NIPS), pages 1565–1573, 2013. 82

[3] 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, pages 611–618, 2001. 82

[4] D. Achlioptas and F. Mcsherry. Fast computation of low-rank matrix approximations. J. ACM, 54(2), April 2007. 82

[5] M. Aghagolzadeh and H. Radha. New guarantees for blind compressed sensing. In Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 1227–1234, 2015. 79

[6] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing, 54(11), 2006. 6, 59, 60, 62

[7] N. Ailon and B. Chazelle. The fast Johnson-Lindenstrauss transform and approximate nearest neighbors. SIAM Journal on Computing, 39:302–322, 2009. 15, 37, 40, 41, 74, 84, 103

[8] N. Ailon, R. Jaiswal, and C. Monteleoni. Streaming k-means approximation. In Advances in Neural Information Processing Systems, pages 10–18, 2009. 143

[9] A. Alaoui and M. Mahoney. Fast randomized kernel methods with statistical guarantees. In Advances in Neural Information Processing Systems, pages 775–783, 2015. 125

[10] Z. Allen-Zhu and Y. Li. LazySVD: Even faster SVD decomposition yet without agonizing pain. In Advances in Neural Information Processing Systems, pages 974–982, 2016. 31

[11] N. Aronszajn. Theory of reproducing kernels. Transactions of the American mathematical society, pages 337–404, 1950. 124

[12] R. Arora, A. Cotter, K. Livescu, and N. Srebro. Stochastic optimization for PCA and PLS. In 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 861–868, 2012. 27

[13] R. Arora, A. Cotter, and N. Srebro. Stochastic optimization of PCA with capped MSG. In Advances in Neural Information Processing Systems (NIPS), pages 1815–1823, 2013. 27

[14] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In SODA, pages 1027–1035, 2007. 101, 104, 107, 131, 144, 146, 147

[15] M. Azizyan, A. Krishnamurthy, and A. Singh. Extreme compressive sampling for covariance estimation. arXiv preprint arXiv:1506.00898, 2015. 2, 79, 93, 97

[16] F. Bach and M. Jordan. Kernel independent component analysis. Journal of machine learning research, 3:1–48, 2002. 120

[17] F. Bach and M. Jordan. Predictive low-rank decomposition for kernel methods. In Proceedings of the 22nd international conference on Machine learning, pages 33–40, 2005. 120

[18] F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In International Conference on Machine Learning, page 6, 2004. 124

[19] M. Balcan, S. Ehrlich, and Y. Liang. Distributed k-means and k-median clustering on general topologies. In Advances in Neural Information Processing Systems (NIPS), pages 1995–2003, 2013. 26

[20] R. Baraniuk, V. Cevher, and M. Wakin. Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective. Proceedings of the IEEE, 98(6):959–971, 2010. 4, 59

[21] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of the restricted isometry property for random matrices. Constructive Approximation, 28(3):253–263, 2008. 15

[22] P. Berkhin. A survey of clustering data mining techniques. In Grouping multidimensional data, pages 25–71. Springer, 2006. 7

[23] M. Berry, D. Mezher, B. Philippe, and A. Sameh. Parallel algorithms for the singular value decomposition. STATISTICS TEXTBOOKS AND MONOGRAPHS, 184:117, 2006. 31

[24] C. Bishop. Pattern recognition and machine learning. Springer, 2006. 7, 82, 99, 130, 131, 142

[25] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. SIAM Journal on Computing, 43(2):687–717, 2014. 109

[26] C. Boutsidis, A. Zouzias, M. Mahoney, and P. Drineas. Randomized dimensionality reduction for k-means clustering. IEEE Transactions on Information Theory, 61(2):1045–1062, 2015. 16, 18, 83, 84, 109, 146

[27] E. Candès and M. Wakin. An introduction to compressive sampling. IEEE Signal Processing Magazine, 25(2):21–30, 2008. 26, 59, 75

[28] V. Chandrasekaran and M. Jordan. Computational and statistical tradeoffs via convex relaxation. Proc. of the National Academy of Sciences, 110(13):E1181–E1190, 2013. 28

[29] C. Chang and C. Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2(3):27:1–27:27, 2011. 122, 139, 148, 155

[30] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. SIAM review, 43(1):129–159, 2001. 5

[31] Y. Chen, Y. Chi, and A. Goldsmith. Exact and stable covariance estimation from quadratic sampling via convex programming. IEEE Trans. on Information Theory, 61:4034–4059, 2015. 28, 79

[32] 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, pages 163–172, 2015. 83

[33] P. Constantine. Active subspaces: Emerging ideas for dimension reduction in parameter studies. SIAM, 2015. 158, 160

[34] P. Constantine, E. Dow, and Q. Wang. Active subspace methods in theory and practice: applications to kriging surfaces. SIAM Journal on Scientific Computing, 36(4):A1500–A1524, 2014. 160

[35] P. Constantine, A. Eftekhari, and M. Wakin. Computing active subspaces efficiently with gradient sketching. In IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pages 353–356, 2015. 163

[36] P. Constantine and D. Gleich. Computing active subspaces with Monte Carlo. arXiv preprint arXiv:1408.0545, 2014. 161

[37] C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In AISTATS, pages 113–120, 2010. 125, 155

[38] C. Cortes and V. Vapnik. Support-vector networks. Machine learning, 20(3):273–297, 1995. 8, 124

[39] J. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. Journal of Machine Learning Research, 16:2859–2900, 2015. 12

[40] S. Dasgupta. The hardness of k-means clustering. Technical Report CS2008-0916, UCSD, 2008. 144

[41] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. Random Structures & Algorithms, 22(1):60–65, 2003. 75

[42] M. Davenport, P. Boufounos, M. Wakin, and R. Baraniuk. Signal processing with compressive measurements. IEEE Journal of Selected Topics in Signal Processing, 4:445–460, 2010. 2, 12, 60, 78, 143

[43] C. Davis and W. Kahan. The rotation of eigenvectors by a perturbation. III. SIAM J. on Numerical Analysis, 7:1–46, 1970. 44, 184

[44] T. Do, L. Gan, N. Nguyen, and T. Tran. Fast and efficient compressive sensing using structurally random matrices. IEEE Transactions on Signal Processing, 60(1):139–154, 2012. 30, 37, 60, 84

[45] D. Donoho. Compressed sensing. IEEE Transactions on Information Theory, 52(4):1289–1306, 2006. 26

[46] P. Drineas, R. Kannan, and M. Mahoney. Fast monte carlo algorithms for matrices I: Approximating matrix multiplication. SIAM Journal on Computing, 36(1):132–157, 2006. 121, 129, 148

[47] P. Drineas, M. Magdon-Ismail, M. Mahoney, and D. Woodruff. Fast approximation of matrix coherence and statistical leverage. The Journal of Machine Learning Research, 13:3475–3506, 2012. 15, 81

[48] P. Drineas and M. Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. The Journal of Machine Learning Research, pages 2153–2175, 2005. 121, 130

[49] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. Psychometrika, 1(3):211–218, 1936. 123

[50] M. Elad. Optimized projections for compressed sensing. IEEE Transactions on Signal Processing, 55:5695–5702, 2007. 37

[51] M. Elad, M. Figueiredo, and Y. Ma. On the role of sparse and redundant representations in image processing. Proceedings of the IEEE, 98(6):972–982, 2010. 59

[52] K. Engan, S. Aase, and J. Husoy. Method of optimal directions for frame design. In IEEE International Conference on Acoustics, Speech and Signal Processing, pages 2443–2446, 1999. 59

[53] P. Evans and M. Annunziata. Industrial internet: Pushing the boundaries. Technical report, General Electric Reports, 2012. 1

[54] J. Fan, F. Han, and H. Liu. Challenges of big data analysis. National science review, 1(2):293–314, 2014. 1

[55] 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, pages 1434–1453, 2013. 83, 143

[56] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. Journal of Machine Learning Research, 2:243–264, 2001. 120

[57] J. Fowler. Compressive-projection principal component analysis. IEEE Transactions on Image Processing, 18(10):2230–2242, 2009. 28, 60, 79

[58] J. Friedman, T. Hastie, and R. Tibshirani. The elements of statistical learning. Springer, 2001. 60

[59] 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, pages 707–717, 2014. 82

[60] A. Gilbert, J. Park, and M. Wakin. Sketched SVD: Recovering spectral features from compressive measurements. arXiv preprint arXiv:1211.0361, 2012. 78, 184

[61] M. Girolami. Mercer kernel-based clustering in feature space. IEEE Transactions on Neural Networks, 13(3):780–784, 2002. 8, 124

[62] A. Gittens and M. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. In International Conference on Machine Learning, pages 567–575, 2013. 121, 130, 132

[63] S. Gleichman and Y. Eldar. Blind compressed sensing. IEEE Transactions on Information Theory, 57(10):6958–6975, 2011. 68, 78

[64] A. Golts and M. Elad. Linearized kernel dictionary learning. IEEE Journal of Selected Topics in Signal Processing, 10(4):726–739, 2016. 126, 138

[65] G. H. Golub and C. F. Van Loan. Matrix computations, volume 3. JHU Press, 2012. 31

[66] M. Gönen and E. Alpaydın. Multiple kernel learning algorithms. Journal of Machine Learning Research, 12:2211–2268, 2011. 124

[67] N. Halko, P. Martinsson, Y. Shkolnisky, and M. Tygert. An algorithm for the principal component analysis of large data sets. SIAM Journal on Scientific Computing, 33(5):2580–2594, 2011. xiv, 27, 48, 49

[68] N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2):217–288, 2011. 12, 15, 27, 75, 76, 84, 109, 120, 126, 144

[69] L. Hogben. Handbook of linear algebra. CRC Press, 2006. 44

[70] Roger A. Horn and Charles R. Johnson. Topics in Matrix Analysis. Cambridge University Press, 1991. 96

[71] C. Hsieh, S. Si, and I. Dhillon. Fast prediction for large-scale kernel machines. In Advances in Neural Information Processing Systems, pages 3689–3697, 2014. 125

[72] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. Journal of the ACM (JACM), 53(3):307–323, 2006. 26

[73] A. Iosifidis and M. Gabbouj. Nyström-based approximate kernel subspace learning. Pattern Recognition, 57:190–197, 2016. 142

[74] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In Proceedings of the forty-fifth annual ACM symposium on Theory of computing, pages 665–674, 2013. 162, 163

[75] Z. Jiang, Z. Lin, and L. Davis. Label consistent K-SVD: Learning a discriminative dictionary for recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(11):2651–2664, 2013. 6

[76] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. Contemporary mathematics, 26:189–206, 1984. 15

[77] I. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. The Annals of Statistics, 29(2):295–327, 2001. 29

[78] I. Jolliffe. Principal component analysis. Springer-Verlag New York, 2 edition, 2002. 25, 74

[79] M. Jordan. On statistics, computation and scalability. Bernoulli, 19(4):1378–1390, 2013. 25

[80] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized power method for sparse principal component analysis. The Journal of Machine Learning Research, 11:517–553, 2010. 38

[81] Tapas Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(7):881–892, Jul 2002. 110

[82] F. Krahmer and R. Ward. New and improved Johnson-Lindenstrauss embeddings via the restricted isometry property. SIAM Journal on Mathematical Analysis, 43(3):1269–1281, 2011. 15

[83] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the Nyström method. Journal of Machine Learning Research, 13:981–1006, 2012. 121, 126, 127, 128, 129, 142

[84] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Ensemble Nyström method. In Advances in Neural Information Processing Systems, pages 1060–1068, 2009. 120, 127

[85] Q. Le, T. Sarlós, and A. Smola. Fastfood-approximating kernel expansions in loglinear time. In Proceedings of the international conference on machine learning, 2013. 85

[86] M. Li, W. Bi, J. Kwok, and B. Lu. Large-scale Nyström kernel matrix approximation using randomized SVD. IEEE transactions on neural networks and learning systems, 26(1):152–164, 2015. 127, 132

[87] 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, pages 287–296, 2006. 14, 26, 30, 33, 34, 79

[88] E. Liberty and S. Zucker. The mailman algorithm: A note on matrix–vector multiplication. Information Processing Letters, 109(3):179–182, 2009. 144

[89] J. Lin and D. Gunopulos. Dimensionality reduction by random projection and latent semantic indexing. In proceedings of the Text Mining Workshop, at the 3rd SIAM Int. Conf. on Data Mining, 2003. 31

[90] X. Liu, C. Aggarwal, Y. Li, X. Kong, X. Sun, and S. Sathe. Kernelized matrix factorization for collaborative filtering. In SIAM Conference on Data Mining, pages 399–416, 2016. 124

[91] S. Lloyd. Least squares quantization in PCM. IEEE transactions on information theory, 28(2):129–137, 1982. 7, 131

[92] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. Large scale kernel machines, pages 301–320, 2007. 115

[93] P. Ma, M. Mahoney, and B. Yu. A statistical perspective on algorithmic leveraging. In Proceedings of the 31st International Conference on Machine Learning (ICML), pages 91–99, 2014. 80

[94] L. Mackey, M. Jordan, R. Chen, B. Farrell, and J. Tropp. Matrix concentration inequalities via the method of exchangeable pairs. The Annals of Probability, 42(3):906–945, 2014. 186

[95] M. Mahoney. Randomized algorithms for matrices and data. Foundations and Trends in Machine Learning, 3(2):123–224, 2011. 12, 26, 27, 75, 80, 126

[96] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(4):791–804, 2012. 59

[97] J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. Foundations and Trends® in Computer Graphics and Vision, 8(2-3):85–283, 2014. 5

[98] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. Bach. Supervised dictionary learning. In Advances in neural information processing systems, pages 1033–1040, 2009. 59

[99] P. Massart. Concentration Inequalities and Model Selection. Springer-Verlag Berlin Heidelberg, 2003. 186

[100] I. Mitliagkas, C. Caramanis, and P. Jain. Memory limited, Streaming PCA. In Advances in Neural Information Processing Systems (NIPS), pages 2886–2894, 2013. xiv, 2, 27, 48, 49, 77, 80

[101] K. Murphy. Machine learning: a probabilistic perspective. The MIT Press, 2012. 7

[102] S. Muthukrishnan. Data streams: Algorithms and applications. Now Publishers Inc, 2005. 74

[103] D. Omidiran and M. Wainwright. High-dimensional variable selection with sparse random projections: measurement sparsity and statistical efficiency. The Journal of Machine Learning Research, 99:2361–2386, 2010. 30

[104] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. J. ACM, 59(6):28, 2012. 144, 146

[105] D. Papailiopoulos, A. Dimakis, and S. Korokythakis. Sparse PCA through low-rank approximations. In Proceedings of The 30th International Conference on Machine Learning (ICML), pages 747–755, 2013. 38

[106] J. Park, M. Wakin, and A. Gilbert. Modal analysis with compressive measurements. IEEE Transactions on Signal Processing, 62(7):1655–1670, 2014. 78

[107] F. Pourkamali-Anaraki. Estimation of the sample covariance matrix from compressive measurements. IET Signal Processing, 10(9):1089–1095, 2016. 22

[108] F. Pourkamali-Anaraki and S. Becker. A randomized approach to efficient kernel clustering. In IEEE Global Conference on Signal and Information Processing, 2016. 10, 126, 138

[109] F. Pourkamali-Anaraki, S. Becker, and S. Hughes. Efficient dictionary learning via very sparse random projections. In Sampling Theory and Applications (SampTA), pages 478–482, 2015. 79, 143

[110] F. Pourkamali-Anaraki and S. Hughes. Compressive K-SVD. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5469–5473, 2013. 60, 61, 62, 79

[111] F. Pourkamali-Anaraki and S. Hughes. Kernel compressive sensing. In IEEE International Conference on Image Processing, pages 494–498, 2013. 9, 124

[112] 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), pages 2332–2336, 2014. 28, 60, 79

[113] 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), pages 1341–1349, 2014. 22, 60, 79, 143

[114] 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), pages 937–940, 2012. 28, 60, 79

[115] H. Raja and W. Bajwa. Cloud K-SVD: Computing data-adaptive representations in the cloud. In 51st Annual Allerton Conf. on Communication, Control, and Computing (Allerton), pages 1474–1481, 2013. 26, 60

[116] H. Raja and W. Bajwa. Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data. IEEE Transactions on Signal Processing, 64(1):173–188, 2016. 2

[117] H. Reboredo, F. Renna, R. Calderbank, and M. Rodrigues. Bounds on the number of measurements for reliable compressive classification. IEEE Transactions on Signal Processing, 64(22):5778–5793, 2016. 12

[118] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Technical report, CS Technion, 2008. 70, 72

[119] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In International Conference on Machine Learning, pages 515–521, 1998. 155

[120] B. Schölkopf and A. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2001. 124

[121] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Neural computation, 10(5):1299–1319, 1998. 8, 124

[122] J. Shawe-Taylor and N. Cristianini. Kernel methods for pattern analysis. Cambridge university press, 2004. 8, 124

[123] M. Shindler, A. Wong, and A. Meyerson. Fast and accurate k-means for large datasets. In Advances in neural information processing systems, pages 2375–2383, 2011. 143

[124] J. Silva, M. Chen, Y. Eldar, G. Sapiro, and L. Carin. Blind compressed sensing over a structured union of subspaces. arXiv preprint arXiv:1103.2469, 2011. 61, 62, 69

[125] V. Sindhwani, T. Sainath, and S. Kumar. Structured transforms for small-footprint deep learning. In Advances in Neural Information Processing Systems, pages 3088–3096, 2015. 85

[126] K. Skretting and K. Engan. Recursive least squares dictionary learning algorithm. IEEE Transactions on Signal Processing, 58(4):2121–2130, 2010. 60

[127] 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., 31(5):18–31, 2014. 25, 74, 83, 125

[128] C. Studer and R. Baraniuk. Dictionary learning from sparsely corrupted or compressed signals. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3341–3344, 2012. 61, 62, 79

[129] S. Sun, J. Zhao, and J. Zhu. A review of Nyström methods for large-scale machine learning. Information Fusion, 26:36–48, 2015. 120, 127, 129, 132, 142

[130] T. Tao. Topics in random matrix theory, volume 132 of Graduate Studies in Mathematics. American Mathematical Society, 2012. 75

[131] P. Traganitis, K. Slavakis, and G. Giannakis. Sketch and validate for big data clustering. IEEE Journal of Selected Topics in Signal Processing, 9(4):678–690, 2015. 83, 143

[132] J. Tropp. Improved analysis of the subsampled randomized Hadamard transform. Advances in Adaptive Data Analysis, pages 115–126, 2011. 85, 103, 143, 193

[133] J. Tropp. An introduction to matrix concentration inequalities. Foundations and Trends® in Machine Learning, 8(1-2):1–230, 2015. 78

[134] J. Tropp and A. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. IEEE Transactions on information theory, 53(12):4655–4666, 2007. 5

[135] J. Tropp and S. Wright. Computational methods for sparse solution of linear inverse problems. Proceedings of the IEEE, 98(6):948–958, 2010. 63

[136] M. Turk and A. Pentland. Face recognition using eigenfaces. In Computer Vision and Pattern Recognition, pages 586–591, 1991. 3

[137] H. Van Nguyen, V. Patel, N. Nasrabadi, and R. Chellappa. Kernel dictionary learning. In IEEE International Conference on Acoustics, Speech and Signal Processing, pages 2021–2024, 2012. 9, 124

[138] H. Van Nguyen, V. Patel, N. Nasrabadi, and R. Chellappa. Design of non-linear kernel dictionaries for object recognition. IEEE Transactions on Image Processing, pages 5123–5135, 2013. 124

[139] R. Vershynin. How close is the sample covariance matrix to the actual covariance matrix? Journal of Theoretical Probability, 25(3):655–686, 2012. 30

[140] S. Wang and Z. Zhang. Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling. Journal of Machine Learning Research, 14(1):2729–2769, 2013. 127, 132

[141] M. Warmuth and D. Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In Advances in neural information processing systems, pages 1481–1488, 2006. 27

[142] R. Willett, M. Duarte, M. Davenport, and R. Baraniuk. Sparsity and structure in hyperspectral imaging: Sensing, reconstruction, and target detection. IEEE signal processing magazine, 31(1), 2014. 3

[143] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In Proceedings of the 14th Annual Conference on Neural Information Processing Systems, pages 682–688, 2001. 121, 126, 127, 129, 148

[144] D. Woodruff. Sketching as a tool for numerical linear algebra. Foundations and Trends® in Theoretical Computer Science, 10(1–2):1–157, 2014. 12

[145] Xindong Wu and et al. Top 10 algorithms in data mining. Knowledge and Information Systems, 14(1):1–37, 2008. 82

[146] M. Yang, L. Zhang, X. Feng, and D. Zhang. Fisher discrimination dictionary learning for sparse representation. In IEEE International Conference on Computer Vision, pages 543–550, 2011. 6

[147] K. Zhang and J. Kwok. Clustered Nyström method for large scale manifold learning and dimension reduction. IEEE Transactions on Neural Networks, 21(10):1576–1587, 2010. 121, 128, 130, 131, 132, 140, 142, 148, 149

[148] K. Zhang, L. Lan, Z. Wang, and F. Moerchen. Scaling up kernel svm on limited resources: A low-rank linearization approach. In AISTATS, pages 1425–1434, 2012. 126, 138

[149] K. Zhang, I. Tsang, and J. Kwok. Improved Nyström low-rank approximation and error analysis. In International conference on Machine learning, pages 1232–1239, 2008. 130

[150] K. Zhang, L. Zhang, and M. Yang. Real-time compressive tracking. In Computer Vision–ECCV, pages 864–877. Springer, 2012. 30

[151] K. Zhang, L. Zhang, and M. Yang. Fast compressive tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(10):2002–2015, 2014. 30, 60, 143

[152] Y. Zhang and L. Ghaoui. Large-scale sparse principal component analysis with application to text data. In Advances in Neural Information Processing Systems (NIPS), pages 532–539, 2011. 38

Appendix A

Appendix to Chapter 2

A.1 Useful Lemmas

Lemma A.1. *Let $\mathbf{R} \in \mathbb{R}^{p\times m}$ be a random matrix consisting of i.i.d. entries with zero mean, finite second and fourth moments μ_2 and μ_4, and kurtosis $\kappa \triangleq \frac{\mu_4}{\mu_2^2} - 3$. We define $\mathbf{t}_k$ as the k^{th} column of the matrix $\mathbf{R}\mathbf{R}^T$. Let $\{\mathbf{e}_i\}_{i=1}^p \in \mathbb{R}^p$ denote the standard basis. If we define the matrix $\Lambda_{k,l} \triangleq \mathbb{E}[\mathbf{t}_k\mathbf{t}_l^T]$, $1 \le k, l \le p$, then*

$$\begin{aligned}\Lambda_{k,k} &= m\mu_2^2\mathbf{I}_{p\times p} + m\mu_2^2\left(\kappa + m + 1\right)\mathbf{e}_k\mathbf{e}_k^T \\ \Lambda_{k,l} &= m^2\mu_2^2\mathbf{e}_k\mathbf{e}_l^T + m\mu_2^2\mathbf{e}_l\mathbf{e}_k^T, \quad k \neq l. \end{aligned} \tag{A.1}$$

Moreover, we have that

$$\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\mathbf{R}\mathbf{R}^T\right] = m\mu_2^2\left(\kappa + m + p + 1\right)\mathbf{I}_{p\times p}. \tag{A.2}$$

Proof. Let r_{ij} denote the i,j^{th} entry of $\mathbf{R}$, thus $\mathbf{t}_k = [\sum_{j=1}^m r_{1j}r_{kj}, \ldots, \sum_{j=1}^m r_{pj}r_{kj}]^T$. We can then calculate the entries of $\Lambda_{k,k}$ and $\Lambda_{k,l}$. For example, the k^{th} diagonal entry of $\Lambda_{k,k}$ is

$$\mathbb{E}[(\sum_{j=1}^m r_{kj}^2)^2] = \mathbb{E}[\sum_{j=1}^m r_{kj}^4 + \sum_{i\neq j} r_{ki}^2 r_{kj}^2] = m\mu_4 + \left(m^2 - m\right)\mu_2^2$$

and its i^{th} diagonal entry for $i \neq k$ is $\mathbb{E}[(\sum_{j=1}^m r_{ij}r_{kj})^2] = \mathbb{E}[\sum_{j=1}^m r_{ij}^2 r_{kj}^2] = m\mu_2^2$, where we used $\mu_1 = 0$.

Moreover, since $\mathbf{R}\mathbf{R}^T$ is a symmetric matrix, the k,l^{th} entry of $\mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{R}\mathbf{R}^T]$ is $\mathbb{E}[\langle \mathbf{t}_k, \mathbf{t}_l\rangle] = \mathbb{E}[\mathrm{tr}(\mathbf{t}_k\mathbf{t}_l^T)] = \mathrm{tr}\left(\Lambda_{k,l}\right)$. □

Lemma A.2. *Assume that* $\mathbf{R} \in \mathbb{R}^{p\times m}$ *is a random matrix consisting of i.i.d. entries drawn from a distribution with zero mean, third moment* $\mu_3 = 0$*, and bounded moments* μ_2*,* μ_4*,* μ_6*, and* μ_8*. Also, let* $\mathbf{x} \in \mathbb{R}^p$ *be a fixed vector. Then*

$$\frac{1}{m^2\,(m+1)^2\,\mu_2^4}\mathbb{E}\Big[\,\big\|\mathbf{R}\mathbf{R}^T\mathbf{x}\big\|_2^4\,\Big] = \xi_1\sum_{i=1}^{p} x_i^4 + \xi_2\sum_{i\neq j} x_i^2 x_j^2 \tag{A.3}$$

where ξ_1 *and* ξ_2 *are given in (2.12) and (2.13).*

Proof. This is a straightforward, albeit tedious, computation. Let $\mathbf{x}=[x_1,\ldots,x_p]^T$ and $\mathbf{t}_k$ denotes the k^{th} column of $\mathbf{R}$. We see that

$$\big\|\mathbf{R}\mathbf{R}^T\mathbf{x}\big\|_2^4 = \Big(\sum_{i=1}^{p}\sum_{j=1}^{p} x_i x_j \langle \mathbf{t}_i, \mathbf{t}_j\rangle\Big)^2.$$

Therefore, similar to Lemma A.1, we need to find $\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_j\rangle\langle \mathbf{t}_k, \mathbf{t}_l\rangle]$ for all possible values of i, j, k, and l. Using the assumption that $\mu_1 = \mu_3 = 0$, we see that the following terms are zero:

$$\begin{aligned}
\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_i\rangle\langle \mathbf{t}_i, \mathbf{t}_j\rangle] &= 0, \quad i \neq j \\
\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_i\rangle\langle \mathbf{t}_j, \mathbf{t}_k\rangle] &= 0, \quad i \neq j \neq k \\
\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_j\rangle\langle \mathbf{t}_i, \mathbf{t}_k\rangle] &= 0, \quad i \neq j \neq k \\
\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_j\rangle\langle \mathbf{t}_k, \mathbf{t}_l\rangle] &= 0, \quad i \neq j \neq k \neq l.
\end{aligned}$$

Next, we can compute the three terms (with $i \neq j$):

$$\begin{aligned}
&\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_i\rangle^2] = m\mu_8 + \mu_6\mu_2\{4m(m-1) + 2m(p-1)\} \\
&+\mu_4^2\{3m(m-1) + m(p-1)\} + \mu_4\mu_2^2\{6m(m-1)(m-2) \\
&+6m(m-1)(p-1) + m(p-1)(p-2)\} \\
&+\mu_2^4\{m(m-1)(m-2)(m-3) + 3m(m-1)(p-1) \\
&+2m(m-1)(m-2)(p-1) + m(m-1)(p-1)(p-2)\}
\end{aligned}$$

$$\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_i\rangle\langle \mathbf{t}_j, \mathbf{t}_j\rangle] = 2m\mu_6\mu_2 + \mu_4^2\{m^2+m\}$$
$$+\mu_4\mu_2^2\{6m(m-1)+2m^2(m-1)+3m(p-2)+2m^2(p-2)\}$$
$$+\mu_2^4\{2m(m-1)(m-2)+3m(m-1)+m^2(m-1)^2$$
$$+3m(m-1)(p-2)+2m^2(m-1)(p-2)+m^2(p-2)(p-3)\}$$

$$\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_j\rangle^2] = 2m\mu_6\mu_2 + 2m\mu_4^2 + \mu_4\mu_2^2\{10m(m-1)$$
$$+5m(p-2)\}+\mu_2^4\{4m(m-1)(m-2)+5m(m-1)(p-2)$$
$$+m(p-2)(p-3)+2m(m-1)\}.$$

Therefore, we see that

$$\frac{1}{m^2\left(m+1\right)^2\mu_2^4}\mathbb{E}\Big[\left\|\mathbf{R}\mathbf{R}^T\mathbf{x}\right\|_2^4\Big] = \xi_1\sum_{i=1}^{p}x_i^4 + \xi_2\sum_{i\neq j}x_i^2x_j^2$$

where

$$\xi_1 = \frac{1}{m^2\left(m+1\right)^2\mu_2^4}\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_i\rangle^2],$$
$$\xi_2 = \frac{1}{m^2\left(m+1\right)^2\mu_2^4}\{\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_i\rangle\langle \mathbf{t}_j, \mathbf{t}_j\rangle] + 2\mathbb{E}[\langle \mathbf{t}_i, \mathbf{t}_j\rangle^2]\}.$$

When the entries of $\mathbf{R}$ are drawn i.i.d. from a Gaussian distribution, we have $\xi_1 = \xi_2$. However, we find an upper bound for a general random matrix $\mathbf{R}$. □

Lemma A.3. *Consider the model for centered data, $\mathbf{x} = \sum_{j=1}^{d} w_j\sigma_j\mathbf{v}_j + \mathbf{z}$, given in Section 2.3. Let us define $\widetilde{h} \triangleq \sum_{j=1}^{d}\sigma_j^4$ and $\beta \triangleq \mathbb{E}\left[w^4\right] - 1$. Then*

$$\mathbb{E}\Big[\left\|\mathbf{x}\right\|_2^4\Big] = h^2\Big[\Big(1+\frac{1}{SNR}\Big)^2 + \Big(\beta\frac{\widetilde{h}}{h^2} + \frac{4}{p}\frac{1}{SNR} + \frac{2}{p}\Big(\frac{1}{SNR}\Big)^2\Big)\Big]. \tag{A.4}$$

Proof. First, note that for centered $\mathbf{x}$, we have

$$\mathbf{x}^T\mathbf{x} = (\sum_{j=1}^{d} w_j\sigma_j\mathbf{v}_j^T + \mathbf{z}^T)(\sum_{j=1}^{d} w_j\sigma_j\mathbf{v}_j + \mathbf{z})$$
$$= \sum_{i=1}^{d}\sum_{j=1}^{d} w_iw_j\sigma_i\sigma_j\mathbf{v}_i^T\mathbf{v}_j + 2\sum_{j=1}^{d} w_j\sigma_j\mathbf{v}_j^T\mathbf{z} + \mathbf{z}^T\mathbf{z}$$
$$= \sum_{i=1}^{d} w_i^2\sigma_i^2 + 2\sum_{j=1}^{d} w_j\sigma_j\mathbf{v}_j^T\mathbf{z} + \mathbf{z}^T\mathbf{z} \tag{A.5}$$

where the last step follows from the fact that $\mathbf{v}_i^T\mathbf{v}_i=1$ and $\mathbf{v}_i^T\mathbf{v}_j=0$ for $i\neq j$. Now, we see that

$$\mathbb{E}\Big[\ \|\mathbf{x}\|_2^4\ \Big] = \mathbb{E}\Big[\Big(\sum_{i=1}^{d} w_i^2\sigma_i^2 + 2\sum_{j=1}^{d} w_j\sigma_j\mathbf{v}_j^T\mathbf{z} + \mathbf{z}^T\mathbf{z}\Big)^2\Big]$$
$$= \sum_{i=1}^{d}\mathbb{E}\Big[w_i^4\Big]\sigma_i^4 + \sum_{i=1}^{d}\sum_{j\neq i}\mathbb{E}\Big[w_i^2\Big]\mathbb{E}\Big[w_j^2\Big]\sigma_i^2\sigma_j^2 + \mathbb{E}\Big[\ \|\mathbf{z}\|_2^4\ \Big]$$
$$+2\sum_{i=1}^{d}\mathbb{E}\Big[w_i^2\Big]\sigma_i^2\mathbb{E}\Big[\mathbf{z}^T\mathbf{z}\Big] + 4\sum_{i=1}^{d}\mathbb{E}\Big[w_i^2\Big]\sigma_i^2\mathbf{v}_i^T\mathbb{E}\Big[\mathbf{z}\mathbf{z}^T\Big]\mathbf{v}_i$$

where we used $\mathbb{E}[w_j]=0$. We also need to find $\mathbb{E}[\|\mathbf{z}\|_2^4]$:

$$\mathbb{E}[\|\mathbf{z}\|_2^4] = \mathbb{E}[(\sum_{i=1}^{p} z_i^2)^2] = \sum_{i=1}^{p}\mathbb{E}[z_i^4] + \sum_{i=1}^{p}\sum_{j\neq i}\mathbb{E}[z_i^2]\mathbb{E}[z_j^2]$$
$$= p\Big(3\frac{\epsilon^4}{p^2}\Big) + (p^2-p)\Big(\frac{\epsilon^2}{p}\Big)^2 = \epsilon^4 + \frac{2}{p}\epsilon^4.$$

This completes the proof. □

A.2 Proof of Theorem 2.1

Since w_j, $\mathbf{z}$, and $\mathbf{R}$ are independent with $\mathbb{E}[w_j]=0$ and $\mathbb{E}[\mathbf{z}]=\mathbf{0}$, we have

$$\mathbb{E}\Big[\mathbf{R}\mathbf{R}^T\mathbf{x}\Big] = \mathbb{E}\Big[\mathbf{R}\mathbf{R}^T\Big(\overline{\mathbf{x}}+\sum_{j=1}^{d} w_j\sigma_j\mathbf{v}_j+\mathbf{z}\Big)\Big] = \mathbb{E}\Big[\mathbf{R}\mathbf{R}^T\Big]\overline{\mathbf{x}}.$$

The diagonal entries of the matrix $\mathbf{R}\mathbf{R}^T$ have the form $\sum_{j=1}^{m} r_{kj}^2$, $1\leq k\leq p$, where r_{ij} denotes the i,j^{th} entry of the matrix $\mathbf{R}$, and the off-diagonal entries have the form $\sum_{j=1}^{m} r_{kj}r_{lj}$, $k\neq l$. The entries of $\mathbf{R}$ are i.i.d. with zero mean and variance μ_2, thus $\mathbb{E}[\mathbf{R}\mathbf{R}^T]=m\mu_2\mathbf{I}_{p\times p}$, and we get $\frac{1}{m\mu_2}\mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{x}]=\overline{\mathbf{x}}$. Theorem 2.1 then follows from linearity and the strong law of large numbers.

A.3 Proof of Theorem 2.2

Consider the center estimator $\widehat{\overline{\mathbf{x}}}_n = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{m\mu_2}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i$. We showed in Theorem 2.1 that this is an unbiased estimator for the true center, i.e. $\mathbb{E}[\widehat{\overline{\mathbf{x}}}_n] = \overline{\mathbf{x}}$. Now, we find the variance:

$$\begin{aligned}
&\mathrm{Var}\left(\widehat{\overline{\mathbf{x}}}_n\right) = \mathbb{E}\Big[\,\Big\|\widehat{\overline{\mathbf{x}}}_n - \overline{\mathbf{x}}\Big\|_2^2\,\Big] \\
&= \frac{1}{n^2}\,\mathrm{tr}\left(\mathbb{E}\Big[\sum_{i=1}^{n}\Big(\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \overline{\mathbf{x}}\Big)\sum_{i=1}^{n}\Big(\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \overline{\mathbf{x}}\Big)^T\Big]\right) \\
&\overset{(a)}{=} \frac{1}{n^2}\,\mathrm{tr}\left(\mathbb{E}\Big[\sum_{i=1}^{n}\Big(\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \overline{\mathbf{x}}\Big)\Big(\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \overline{\mathbf{x}}\Big)^T\Big]\right) \\
&= \frac{1}{n}\,\mathrm{tr}\left(\mathbb{E}\Big[\Big(\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\mathbf{x} - \overline{\mathbf{x}}\Big)\Big(\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\mathbf{x} - \overline{\mathbf{x}}\Big)^T\Big]\right)
\end{aligned}$$

where (a) follows from the fact that the vectors $(\frac{1}{m\mu_2}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i - \overline{\mathbf{x}})$ are independent and identically distributed around $\mathbf{0}$ for each i. Also, note that we have dropped the i subscript in the last step, since all $\mathbf{R}_i$ and $\mathbf{x}_i$ are i.i.d. and the dependence on i no longer matters. Expanding $\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\mathbf{x} - \overline{\mathbf{x}}$ as $\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T(\mathbf{x} - \overline{\mathbf{x}}) - (\overline{\mathbf{x}} - \frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\overline{\mathbf{x}})$, we obtain:

$$\begin{aligned}
\mathrm{Var}\Big(\widehat{\overline{\mathbf{x}}}_n\Big) =& \frac{1}{n}\frac{1}{m^2\mu_2^2}\underbrace{\mathrm{tr}\left(\mathbb{E}\Big[\mathbf{R}\mathbf{R}^T\Big(\mathbf{x} - \overline{\mathbf{x}}\Big)\Big(\mathbf{x} - \overline{\mathbf{x}}\Big)^T\mathbf{R}\mathbf{R}^T\Big]\right)}_{=A_1} \\
&+\frac{2}{n}\,\mathrm{tr}\left(\mathbb{E}\Big[\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\Big(\mathbf{x} - \overline{\mathbf{x}}\Big)\Big(\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\overline{\mathbf{x}} - \overline{\mathbf{x}}\Big)^T\Big]\right) \\
&+\frac{1}{n}\underbrace{\mathrm{tr}\left(\mathbb{E}\Big[\Big(\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\overline{\mathbf{x}} - \overline{\mathbf{x}}\Big)\Big(\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\overline{\mathbf{x}} - \overline{\mathbf{x}}\Big)^T\Big]\right)}_{=A_3}. \qquad \text{(A.6)}
\end{aligned}$$

The second term in (A.6) is zero because $\mathbb{E}[\mathbf{x} - \overline{\mathbf{x}}] = \mathbf{0}$. Thus, the variance consists of two terms. The first term in (A.6) is computed using the results of Lemma A.1 as

$$\begin{aligned}
A_1 \;&= m\mu_2^2\,(\kappa + m + p + 1)\,\mathbb{E}\Big[||\mathbf{x} - \overline{\mathbf{x}}||^2\Big] \\
&= m\mu_2^2 h\,(\kappa + m + p + 1)\left(1 + \frac{1}{\mathrm{SNR}}\right) \qquad \text{(A.7)}
\end{aligned}$$

where the last step follows from taking the expectation of Eq. A.5. Next, we find the third term in (A.6):

$$A_3 = \mathbb{E}\left[\left\|\frac{1}{m\mu_2}\mathbf{R}\mathbf{R}^T\overline{\mathbf{x}}-\overline{\mathbf{x}}\right\|_2^2\right] \overset{(b)}{=} \frac{1}{m^2\mu_2^2}\overline{\mathbf{x}}^T\mathbb{E}\big[\mathbf{R}\mathbf{R}^T\mathbf{R}\mathbf{R}^T\big]\overline{\mathbf{x}} - \|\overline{\mathbf{x}}\|_2^2$$
$$\overset{(c)}{=} \frac{1}{m}\left(\kappa+p+1\right)\|\overline{\mathbf{x}}\|_2^2 \tag{A.8}$$

where in (b) we have used $\mathbb{E}[\mathbf{R}\mathbf{R}^T]=m\mu_2\mathbf{I}_{p\times p}$ and (c) follows from Lemma A.1. By substituting (A.7) and (A.8) in (A.6), we get the variance of our center estimator.

A.4 Proof of Theorem 2.3

Consider the model for centered data, $\mathbf{x}=\sum_{j=1}^d w_j\sigma_j\mathbf{v}_j+\mathbf{z}$, given in Section 2.3. Since w_j, $\mathbf{z}$, and $\mathbf{R}$ are independent with $\mathbb{E}[w_j]=0$, $\mathbb{E}[w_j^2]=1$, and $\mathbb{E}[\mathbf{z}]=\mathbf{0}$, it is easy to check that

$$\mathbb{E}\Big[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T\Big] = \sum_{j=1}^d \sigma_j^2\mathbf{C}_{\mathbf{v}}^{(j)} + \mathbf{C}_\epsilon \tag{A.9}$$

where we have defined $\mathbf{C}_{\mathbf{v}}^{(j)} \triangleq \mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{v}_j\mathbf{v}_j^T\mathbf{R}\mathbf{R}^T]$ for $j=1,\ldots,d$, and $\mathbf{C}_\epsilon \triangleq \mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{z}\mathbf{z}^T\mathbf{R}\mathbf{R}^T]$. We first compute the covariance matrix $\mathbf{C}_{\mathbf{v}}^{(j)}$. For simplicity and without loss of generality, we omit the dependence on j. Consider a principal component $\mathbf{v}=[v_1,\ldots,v_p]^T$ with unit norm. Then,

$$\mathbf{C}_{\mathbf{v}} = \mathbb{E}\Big[\mathbf{R}\mathbf{R}^T\mathbf{v}\mathbf{v}^T\mathbf{R}\mathbf{R}^T\Big] \overset{(a)}{=} \mathbb{E}\Big[\Big(\sum_{k=1}^p v_k\mathbf{t}_k\Big)\Big(\sum_{l=1}^p v_l\mathbf{t}_l\Big)^T\Big]$$
$$= \sum_{k=1}^p v_k^2\Lambda_{k,k} + \sum_{k\neq l} v_kv_l\Lambda_{k,l}$$
$$\overset{(b)}{=} \Big(m^2+m\Big)\mu_2^2\mathbf{v}\mathbf{v}^T + m\mu_2^2\mathbf{I}_{p\times p} + m\mu_2^2\kappa\mathrm{diag}(\mathbf{v}\mathbf{v}^T)$$

where in (a) we have defined $\mathbf{t}_k$ as the k^{th} column of $\mathbf{R}\mathbf{R}^T$, in (b) we have used Lemma A.1, and again the notation $\mathrm{diag}(\mathbf{A})$ indicates the matrix formed by zeroing all but the diagonal entries of $\mathbf{A}$. Hence,

$$\mathbf{C}_{\mathbf{v}}^{(j)} = \big(m^2+m\big)\mu_2^2\mathbf{v}_j\mathbf{v}_j^T + m\mu_2^2\mathbf{I}_{p\times p} + m\mu_2^2\kappa\mathrm{diag}(\mathbf{v}_j\mathbf{v}_j^T) \tag{A.10}$$

Next, we find the covariance matrix $\mathbf{C}_\epsilon$ induced by the noise:

$$\mathbf{C}_\epsilon = \mathbb{E}\big[\mathbf{R}\mathbf{R}^T \mathbb{E}_{\mathbf{z}}\big[\mathbf{z}\mathbf{z}^T\big]\mathbf{R}\mathbf{R}^T\big] \overset{(c)}{=} \frac{\epsilon^2}{p} m\mu_2^2(\kappa+m+p+1)\mathbf{I}_{p\times p} \tag{A.11}$$

where (c) follows from $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \frac{\epsilon^2}{p}\mathbf{I}_{p\times p})$ and Lemma A.1. We substitute the covariance matrices in (A.10) and (A.11) into (A.9):

$$\frac{1}{(m^2+m)\,\mu_2^2}\mathbb{E}\Big[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T\Big] = \widehat{\mathbf{C}}_{true} + \mathbf{E} \tag{A.12}$$

where $\widehat{\mathbf{C}}_{true} \triangleq \sum_{j=1}^{d} \sigma_j^2 \mathbf{v}_j\mathbf{v}_j^T + \alpha\mathbf{I}_{p\times p}$, $\alpha \triangleq \frac{h}{m+1} + (\frac{\kappa}{p(m+1)} + \frac{(m+p+1)}{p(m+1)})\epsilon^2$, and $\mathbf{E} \triangleq \frac{\kappa}{m+1}\sum_{j=1}^{d}\sigma_j^2 \mathrm{diag}(\mathbf{v}_j\mathbf{v}_j^T)$.

The rest of Theorem 2.3 then follows from linearity and the strong law of large numbers.

A.5 Proof of Theorem 2.4

Let us define the scaled projected data sample $\mathbf{g}_i \triangleq (1/\sqrt{m\,(m+1)\,\mu_2^2})\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i \in \mathbb{R}^p$. Then, $(\widehat{\mathbf{C}}_n - \mathbf{C}_\infty)$ can be expressed as a sum of independent random matrices:

$$\widehat{\mathbf{C}}_n - \mathbf{C}_\infty = \frac{1}{n}\sum_{i=1}^{n}\mathbf{S}_i, \text{ where } \mathbf{S}_i \triangleq \left(\mathbf{g}_i\mathbf{g}_i^T - \mathbf{C}_\infty\right). \tag{A.13}$$

Based on Theorem 2.3, $\mathbb{E}[\mathbf{g}\mathbf{g}^T]=\mathbf{C}_\infty$, thus for each individual term $\mathbb{E}[\mathbf{S}_i]=\mathbf{0}$. Hence

$$\mathbb{E}\Big[\,\Big\|\widehat{\mathbf{C}}_n - \mathbf{C}_\infty\Big\|_F^2\,\Big] = \frac{1}{n^2}\,\mathrm{tr}\Big(\mathbb{E}\Big[\Big(\sum_{i=1}^{n}\mathbf{S}_i\Big)^2\Big]\Big) = \frac{1}{n}\,\mathrm{tr}\Big(\mathbb{E}\Big[\mathbf{S}^2\Big]\Big)$$
$$= \frac{1}{n}\,\mathrm{tr}\left(\mathbb{E}\Big[\mathbf{g}\mathbf{g}^T\mathbf{g}\mathbf{g}^T\Big] - \mathbf{C}_\infty^2\right) \overset{(a)}{\leq} \frac{1}{n}\mathbb{E}\Big[\,\|\mathbf{g}\|_2^4\,\Big] \overset{(b)}{\leq} \frac{1}{n}\xi\mathbb{E}\Big[\,\|\mathbf{x}\|_2^4\,\Big]$$

where (a) and (b) follow from $\mathrm{tr}(\mathbf{C}_\infty^2) > 0$ and Lemma A.2, respectively. Finally, we use Lemma A.3 to complete the proof.

A.6 Proof of Theorem 2.5

Consider the eigendecomposition of $\widehat{\mathbf{C}}_{true}$ and the perturbed matrix $\widehat{\mathbf{C}}_n$:

$$\widehat{\mathbf{C}}_{true} = \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{bmatrix}$$
$$\widehat{\mathbf{C}}_n = \begin{bmatrix} \widetilde{\mathbf{V}}_1 & \widetilde{\mathbf{V}}_2 \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{S}}_1 & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{S}}_2 \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{V}}_1^T \\ \widetilde{\mathbf{V}}_2^T \end{bmatrix} \tag{A.14}$$

where $\mathbf{V}_1, \widetilde{\mathbf{V}}_1 \in \mathbb{R}^{p\times d'}$, $d' \leq d$, are the first d' true and estimated PCs. Also note $\lambda_i(\widehat{\mathbf{C}}_{true}) = \sigma_i^2 + \alpha$ for $i = 1, \ldots, d$, and $\lambda_i(\widehat{\mathbf{C}}_{true}) = \alpha$, $i = d+1, \ldots, p$. The distance between each perturbed eigenvalue and the corresponding original eigenvalue depends on the amount of perturbation. From Eq. 2.14, we now have that $\left|\lambda_j(\widehat{\mathbf{C}}_n) - \lambda_j(\widehat{\mathbf{C}}_{true})\right| \leq (\varepsilon + \delta)h$ for all $j = 1, \ldots, p$, with probability at least $1 - \frac{1}{n\varepsilon^2}\tau$.

Moreover, it is possible to quantify the rotation of eigenvectors using the notion of canonical angle matrix defined in [43]. The canonical angles between the first d' estimated and true PCs are defined as $\theta_i = \arccos \rho_i$, where $\{\rho_i\}_{i=1}^{d'}$ are the singular values of $(\widetilde{\mathbf{V}}_1^T\widetilde{\mathbf{V}}_1)^{-1/2}\widetilde{\mathbf{V}}_1^T\mathbf{V}_1(\mathbf{V}_1^T\mathbf{V}_1)^{-1/2}$, in our case, just $\widetilde{\mathbf{V}}_1^T\mathbf{V}_1$. The canonical angle matrix is then defined as $\Theta(\widetilde{\mathbf{V}}_1, \mathbf{V}_1) = \mathrm{diag}(\theta_1, \ldots, \theta_{d'})$. Based on the results given in [43, 60], we have $\left\|\sin\Theta(\widetilde{\mathbf{V}}_1, \mathbf{V}_1)\right\|_2 \leq (\varepsilon + \delta)h/\eta$, where $\eta \triangleq \min_{1\leq i\leq d', 1\leq j\leq p-d'} \left|(\mathbf{S}_1)_{ii} - (\widetilde{\mathbf{S}}_2)_{jj}\right| > 0$, which represents the absolute separation between eigenvalues. For any $d' < d$, we have

$$\eta = \left|\lambda_{d'}\left(\widehat{\mathbf{C}}_{true}\right) - \lambda_{d'+1}\left(\widehat{\mathbf{C}}_n\right)\right| = \left|\left(\sigma_{d'}^2 + \alpha\right) - \lambda_{d'+1}\left(\widehat{\mathbf{C}}_n\right)\right|.$$

We find a lower bound for η using the above results on perturbation of eigenvalues:

$$\left|\left(\sigma_{d'+1}^2 + \alpha\right) - \lambda_{d'+1}\left(\widehat{\mathbf{C}}_n\right)\right| \leq (\varepsilon + \delta)\, h.$$

Provided that $(\sigma_{d'}^2 - \sigma_{d'+1}^2) > (\varepsilon + \delta)h$, we get $\eta \geq (\sigma_{d'}^2 - \sigma_{d'+1}^2) - (\varepsilon + \delta)h > 0$. Therefore, with probability at least $1 - \frac{1}{n\varepsilon^2}\tau$, the maximum canonical angle between subspaces θ_{max} satisfies

$$\sin\theta_{max} = \left\|\sin\Theta(\widetilde{\mathbf{V}}_1, \mathbf{V}_1)\right\|_2 \leq \frac{(\varepsilon + \delta)}{\frac{\left(\sigma_{d'}^2 - \sigma_{d'+1}^2\right)}{h} - (\varepsilon + \delta)}.$$

For $d' = d$, we have $\lambda_{d'+1}(\widehat{C}_{true}) = \alpha$, instead of $\sigma^2_{d'+1} + \alpha$ for $d' < d$. Hence, provided that $\sigma^2_d > (\varepsilon + \delta)h$, we have $\eta \geq \sigma^2_d - (\epsilon + \delta)h$ for this case. This completes the proof.

A.7 Proof of Theorem 2.6

Based on Lemma A.1, we compute the expectation $\mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T]$ over the randomness in $\mathbf{R}$ for a fixed data sample $\mathbf{x} = [x_1, \ldots, x_p]^T$:

$$\begin{aligned}\mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T] &= \mathbb{E}[(\sum_{k=1}^{p} x_k\mathbf{t}_k)(\sum_{l=1}^{p} x_l\mathbf{t}_l)^T] \\ &= \left(m^2+m\right)\mu_2^2\mathbf{x}\mathbf{x}^T + \kappa m\mu_2^2\,\mathrm{diag}\left(\mathbf{x}\mathbf{x}^T\right) \\ &\quad + m\mu_2^2\,\mathrm{tr}\left(\mathbf{x}\mathbf{x}^T\right)\mathbf{I}_{p\times p} \end{aligned} \tag{A.15}$$

where we used $\|\mathbf{x}\|_2^2 = \mathrm{tr}(\mathbf{x}\mathbf{x}^T)$. We rescale the expectation:

$$\begin{aligned}&\frac{1}{\left(m^2+m\right)\mu_2^2}\mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T] \\ &= \mathbf{x}\mathbf{x}^T + \frac{\kappa}{m+1}\,\mathrm{diag}\left(\mathbf{x}\mathbf{x}^T\right) + \frac{1}{m+1}\,\mathrm{tr}\left(\mathbf{x}\mathbf{x}^T\right)\mathbf{I}_{p\times p}. \end{aligned} \tag{A.16}$$

Next, we rewrite the sample covariance matrix $\widehat{\mathbf{C}}_n$ as:

$$\widehat{\mathbf{C}}_n = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\left(m^2+m\right)\mu_2^2}\mathbf{R}_i\mathbf{R}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{R}_i\mathbf{R}_i^T.$$

Using linearity of expectation, it is straightforward to see:

$$\mathbb{E}[\widehat{\mathbf{C}}_n] = \mathbf{C}_n + \frac{\kappa}{m+1}\,\mathrm{diag}\left(\mathbf{C}_n\right) + \frac{1}{m+1}\,\mathrm{tr}\left(\mathbf{C}_n\right)\mathbf{I}_{p\times p} \tag{A.17}$$

and this completes the proof.

Appendix B

Appendix to Chapter 4

B.1 Exponential Concentration Inequalities

Below are standard inequalities listed in expedient formats.

Theorem B.1 (Hoeffding's Inequality [99]). *Let $z_1, \ldots, z_n$ be independent, centered random variables, and assume that each one is bounded:*

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

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

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

Theorem B.2 (Bernstein Inequality [99]). *Let $z_1, \ldots, z_n$ be independent, centered random variables, and assume that each one is uniformly bounded:*

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

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 $t \geq 0$:

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

Theorem B.3 (Matrix Bernstein Inequality [94]). *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:*

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

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* $t \geq 0$*:*

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

B.2 Properties of the Sampling Matrix

Theorem B.4. *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:*

$$\mathbb{E}[\mathbf{R}\mathbf{R}^T] = \frac{m}{p}\mathbf{I}_p \tag{B.1}$$

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

$$\mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T] = \frac{m(m-1)}{p(p-1)}\mathbf{x}\mathbf{x}^T + \frac{m(p-m)}{p(p-1)}\operatorname{diag}(\mathbf{x}\mathbf{x}^T). \tag{B.2}$$

Proof. The columns of $\mathbf{R}$ are distinct canonical basis vectors, thus $\mathbf{R}^T\mathbf{R} = \mathbf{I}_m$. To prove (B.1), 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

$$\mathbb{E}[\mathbf{r}_i\mathbf{r}_i^T] = \frac{1}{p}\mathbf{I}_p \text{ for } i = 1, \ldots, m. \tag{B.3}$$

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,

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

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 $r_i = j$, $1 \le j \le p$, 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 (B.1).

Next, we show that (B.2) holds. Note that:

$$\begin{aligned}
&\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T\right]\\
&= \sum_{(r_1,\ldots,r_m)} \mathbb{P}\left\{\mathbf{r}_1 = \mathbf{e}_{r_1},\ldots,\mathbf{r}_m = \mathbf{e}_{r_m}\right\}\times\Big(\sum_{i=1}^m \mathbf{e}_{r_i}\mathbf{e}_{r_i}^T\Big)\mathbf{x}\mathbf{x}^T\Big(\sum_{i=1}^m \mathbf{e}_{r_i}\mathbf{e}_{r_i}^T\Big)\\
&= \Big(\frac{1}{p}\frac{1}{p-1}\cdots\frac{1}{p-(m-1)}\Big)\left\{\alpha_1\sum_{k=1}^p \mathbf{e}_k\mathbf{e}_k^T\mathbf{x}\mathbf{x}^T\mathbf{e}_k\mathbf{e}_k^T + \alpha_2\sum_{k\neq l}\mathbf{e}_k\mathbf{e}_k^T\mathbf{x}\mathbf{x}^T\mathbf{e}_l\mathbf{e}_l^T\right\} \qquad \text{(B.4)}
\end{aligned}$$

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, $1 \le k \le p$, 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:

$$\mathbb{E}\left[\mathbf{R}\mathbf{R}^T\mathbf{x}\mathbf{x}^T\mathbf{R}\mathbf{R}^T\right] = \left(\frac{1}{p}\frac{1}{p-1}\cdots\frac{1}{p-(m-1)}\right)$$
$$\times\left(\alpha_2\Big(\sum_{k=1}^{p}\mathbf{e}_k\mathbf{e}_k^T\Big)\mathbf{x}\mathbf{x}^T\Big(\sum_{k=1}^{p}\mathbf{e}_k\mathbf{e}_k^T\Big) + (\alpha_1 - \alpha_2)\,\mathrm{diag}\left(\mathbf{x}\mathbf{x}^T\right)\right)$$

and this completes the proof. □

Lemma B.1. *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 B.4. For a matrix $\mathbf{R} \in \mathbb{R}^{p\times m}$ containing m distinct canonical basis vectors, we show that $\mathbb{E}[\mathbf{R}\mathbf{R}^T] = \frac{m}{p}\mathbf{I}_p$. Each component of the sub-sampled data $\mathbf{w} = \mathbf{R}\mathbf{R}^T\mathbf{x}$ 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 $\mathbb{E}[\mathbf{w}] = \mathbb{E}[\mathbf{R}\mathbf{R}^T\mathbf{x}] = \frac{m}{p}\mathbf{x}$, we conclude that $\pi_j = \frac{m}{p}$ for $j = 1, \ldots, p$. □

B.3 Proof of Theorem 4.3

We present the proof of Thm. 4.3 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. B.4, we compute the following expectations:

$$\mathbb{E}[\widehat{\mathbf{C}}_{\text{emp}}] = \mathbf{C}_{\text{emp}} + \frac{(p-m)}{(m-1)}\,\mathrm{diag}(\mathbf{C}_{\text{emp}}) \tag{B.5}$$

and

$$\mathbb{E}[\mathrm{diag}(\widehat{\mathbf{C}}_{\text{emp}})] = \mathrm{diag}(\mathbb{E}[\widehat{\mathbf{C}}_{\text{emp}}]) = \frac{(p-1)}{(m-1)}\,\mathrm{diag}(\mathbf{C}_{\text{emp}}). \tag{B.6}$$

Hence, using (B.5) and (B.6), 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. B.3). Note that $(\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}})$ can be written as a sum of n independent centered random matrices:

$$\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}} = \sum_{i=1}^{n} \mathbf{Z}_i = \sum_{i=1}^{n} \frac{1}{n}(\mathbf{Z}_i^{(1)} - \mathbf{Z}_i^{(2)} - \mathbf{Z}_i^{(3)}) \tag{B.7}$$

where $\mathbf{Z}_i^{(1)} = \frac{p(p-1)}{m(m-1)}\mathbf{w}_i\mathbf{w}_i^T$, $\mathbf{Z}_i^{(2)} = \frac{p(p-m)}{m(m-1)}\operatorname{diag}(\mathbf{w}_i\mathbf{w}_i^T)$, $\mathbf{Z}_i^{(3)} = \mathbf{x}_i\mathbf{x}_i^T$, 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 $\mathbf{Z}_i^{(1)}$:

$$\begin{aligned}\|\mathbf{Z}_i^{(1)}\|_2 &= \frac{p(p-1)}{m(m-1)}\|\mathbf{w}_i\mathbf{w}_i^T\|_2 = \frac{p(p-1)}{m(m-1)}\|\mathbf{w}_i\|_2^2 \\ &\leq \frac{p(p-1)}{m(m-1)}\rho\|\mathbf{x}_i\|_2^2 \leq \frac{p(p-1)}{m(m-1)}\rho\|\mathbf{X}\|_{\text{max-col}}^2.\end{aligned} \tag{B.8}$$

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

$$\|\mathbf{Z}_i^{(2)}\|_2 = \frac{p(p-m)}{m(m-1)}\|\operatorname{diag}(\mathbf{w}_i\mathbf{w}_i^T)\|_2 \leq \frac{p(p-m)}{m(m-1)}\|\mathbf{X}\|_{\text{max}}^2. \tag{B.9}$$

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

$$\|\mathbf{Z}_i^{(3)}\|_2 = \|\mathbf{x}_i\mathbf{x}_i^T\|_2 = \|\mathbf{x}_i\|_2^2 \leq \|\mathbf{X}\|_{\text{max-col}}^2 \tag{B.10}$$

and the triangle inequality for the spectral norm leads to:

$$\|\mathbf{Z}_i\|_2 \leq \frac{1}{n}\left(\|\mathbf{Z}_i^{(1)}\|_2 + \|\mathbf{Z}_i^{(2)}\|_2 + \|\mathbf{Z}_i^{(3)}\|_2\right) \leq L \tag{B.11}$$

where L is given in (4.26).

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

$$\sigma^2 = \|\mathbb{E}[(\widehat{\mathbf{C}}_n - \mathbf{C}_{\text{emp}})^2]\|_2 = \|\sum_{i=1}^{n}\mathbb{E}[\mathbf{Z}_i^2]\|_2. \tag{B.12}$$

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

$$\mathbb{E}[\mathbf{Z}_i^2] = \frac{1}{n^2}\mathbb{E}[(\mathbf{Z}_i^{(1)} - \mathbf{Z}_i^{(2)} - \mathbf{Z}_i^{(3)})^2] \tag{B.13}$$

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:

$$\begin{aligned}
&\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(1)}] = \frac{p^2(p-1)^2}{m^2(m-1)^2}\mathbb{E}[\mathbf{w}_i\mathbf{w}_i^T\mathbf{w}_i\mathbf{w}_i^T] \\
&= \frac{p^2(p-1)^2}{m^2(m-1)^2}\mathbb{E}[\|\mathbf{w}_i\|_2^2\mathbf{w}_i\mathbf{w}_i^T] \preccurlyeq \frac{p^2(p-1)^2}{m^2(m-1)^2}\rho\|\mathbf{x}_i\|_2^2\mathbb{E}[\mathbf{w}_i\mathbf{w}_i^T] \\
&= \frac{p(p-1)}{m(m-1)}\rho\|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T + \frac{p(p-1)(p-m)}{m(m-1)^2}\rho\|\mathbf{x}_i\|_2^2\,\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T)
\end{aligned}$$

where the inequality follows the fact that expectation preserves the semidefinite order and we also used Thm. B.4 to compute $\mathbb{E}[\mathbf{w}_i\mathbf{w}_i^T]$. Now, we compute $\mathbb{E}[\mathbf{Z}_i^{(2)}\mathbf{Z}_i^{(2)}]$:

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

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 B.1, 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:

$$\begin{aligned}
&\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(3)}] = \mathbb{E}[\mathbf{Z}_i^{(3)}\mathbf{Z}_i^{(1)}]^T = \frac{p(p-1)}{m(m-1)}\mathbb{E}[\mathbf{w}_i\mathbf{w}_i^T]\mathbf{x}_i\mathbf{x}_i^T \\
&= \|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T + \frac{(p-m)}{(m-1)}\,\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T)\mathbf{x}_i\mathbf{x}_i^T
\end{aligned}$$

and

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

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:

$$\begin{aligned}\sigma^2 = &\Big\|\sum_{i=1}^n \mathbb{E}[\mathbf{Z}_i^2]\Big\|_2 \leq \frac{1}{n}\left(\frac{p(p-1)}{m(m-1)}\rho - 1\right)\cdot\Big\|\frac{1}{n}\sum_{i=1}^n \|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T\Big\|_2 \\ &+\frac{1}{n}\frac{p(p-1)(p-m)}{m(m-1)^2}\rho\Big\|\frac{1}{n}\sum_{i=1}^n \|\mathbf{x}_i\|_2^2\,\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T)\Big\|_2 \\ &+\frac{1}{n^2}\frac{p(p-m)^2}{m(m-1)^2}\Big\|\sum_{i=1}^n(\mathrm{diag}(\mathbf{x}_i\mathbf{x}_i^T))^2\Big\|_2 \\ &+\frac{1}{n^2}\left(\big\|\sum_{i=1}^n \mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}]\big\|_2 + \big\|\sum_{i=1}^n \mathbb{E}[\mathbf{Z}_i^{(2)}\mathbf{Z}_i^{(1)}]\big\|_2\right).\end{aligned}$$

We also have the following two inequalities:

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

and

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

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

$$\begin{aligned}&\Big\|\sum_{i=1}^n \mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}]\Big\|_2 \leq \sum_{i=1}^n \|\mathbb{E}[\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}]\|_2 \\ &\leq \sum_{i=1}^n \mathbb{E}[\|\mathbf{Z}_i^{(1)}\mathbf{Z}_i^{(2)}\|_2] \leq \sum_{i=1}^n \mathbb{E}[\|\mathbf{Z}_i^{(1)}\|_2\|\mathbf{Z}_i^{(2)}\|_2]\end{aligned}$$

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 $\|\mathbf{AB}\|_2 \leq \|\mathbf{A}\|_2\|\mathbf{B}\|_2$. 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

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

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

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

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

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

and this completes the proof.

B.4 Preservation of Pairwise Distances

Theorem B.5. *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* **HD** (4.1) *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 $1 - 3\beta^{-1}$,*

$$0.40\,\|\mathbf{x}_1-\mathbf{x}_2\|_2 \leq \left\| \sqrt{\frac{p}{m}}\mathbf{R}^T\mathbf{HD}(\mathbf{x}_1-\mathbf{x}_2) \right\|_2 \leq 1.48\,\|\mathbf{x}_1-\mathbf{x}_2\|_2 \tag{B.14}$$

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

Proof. This result is a straightforward consequence of Theorem 3.1 in [132]. Let us denote $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$ and represent it as $\mathbf{x} = \mathbf{V}\mathbf{c}$, 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 $\|\mathbf{R}^T\mathbf{HDx}\|_2 = \|\mathbf{R}^T\mathbf{HDVc}\|_2$:

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

where σ_1 and σ_β denote the largest and smallest singular values. Based on [132], for $m \geq 4[\sqrt{\beta} + \sqrt{8\log(\beta p)}]^2\log(\beta)$ and with probability at least $1 - 3\beta^{-1}$,

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

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