Randomized Algorithms for Machine Learning

CHEN, Xixian

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
July 2018

Abstract of thesis entitled:

Randomized Algorithms for Machine Learning

Submitted by CHEN, Xixian

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in July 2018

The power of machine learning techniques has benefited from the explosion of available information involved in big data, whereas the solution of learning problems is typically computationally intractable in the presence of the rapidly increasing data volume. Therefore, the development of approaches to make learning efficient is urgently needed. In recent years, randomized algorithms have received much attention for facilitating computation and yielding approximate outputs for many learning problems. They utilize randomization, including random projection and random sampling, to create a compact data representation for the learning problems, so that a little learning accuracy is sacrificed, while high computational efficiency is achieved. In this thesis, we develop randomized algorithms for three fundamental machine learning problems, and improve the tradeoff between the efficiency in the computation and the accuracy of the approximate solution.

The central contribution of this thesis is divided into three parts. In the first part, we propose a training-efficient feature map to approximate the kernel matrix and accelerate the kernel learning, in which we rely on random projection to derive a fast data-dependent subspace embedding for feature generation. We describe two algorithms with different tradeoffs regarding the running speed and accuracy, and prove that the $O(\ell)$ features they induce are able to perform as accurately as the $O(\ell^2)$ features of other feature map methods.

In the second part, we propose a faster unsupervised online hashing by leveraging fast random projection to transform data in a more compact manner in the PCA (Principal Component Analysis) based online hashing. In particular, we derive independent transformations to guarantee the sketching accuracy, and design a novel implementation to make such transformations applicable to the current PCA based online hashing without increasing the space cost. We rigorously prove that our method can yield a comparable learning accuracy with less time complexity and an equal space cost.

In the final part, we present an efficient and accurate covariance matrix estimation for which we leverage a data-aware sampling method to construct low-dimensional data. We rigorously prove that our proposed estimator is unbiased and requires fewer data to achieve the same accuracy with specially designed sampling distributions compared with all other methods. In addition, we depict that the computational procedures in our algorithm are efficient regarding communication, storage, and calculation time. All achievements imply an improved tradeoff between the estimation accuracy and computational costs.

論文摘要：
標題：隨機算法在大規模機器學習的應用研究
提交者：陳錫顯
學位：博士
香港中文大學，二零壹八年二月

大數據所蘊含的豐富信息給予了機器學習强大的學習能力，然而不斷增長的大數據體積使得機器學習問題的求解計算愈加困難。因此，設計出適當的方法達到有效快速的學習變的非常重要。近些年，隨機算法獲得了大量的關註，因為它有助於快速計算同時能夠獲得和原來求解問題的相似的結果。隨機算法的核心思想是使用隨機化包括隨機投影和隨機抽樣，為原來的學習問題構造出壹個更加緊湊的數據表達形式。在本論文裏，我們主要為三種根本的機器學習問題設計了相應的隨機算法，同時提高求解效率和結果準確度上的平衡。

本篇論文的主要貢獻可以分為三個部分。在第壹個部分，我們提出了壹種特征生成方法用來近似核矩陣進而加快核學習的訓練。其中，我們依賴隨機投影去設計壹種快速的數據敏感的子空間嵌入方法用來產生最終所需的特征。我們描述了兩種在計算效率和結果準確度上不同平衡的兩種算法，並且都證明了相對於其他方法我們的方法可以用少很多的特征數量去達到同等規模的結果準確度。

在第二個部分，我們提出了壹種更快的無監督在線的哈希方法，主要是通過使用壹種快速的隨機投影來壓縮數據並且用於基於主成分分析的在線哈希上。特別的，我們提出使用獨立同分布的投影方式來保證壓縮的準確性，同時設計出有效的實現方式使得這種投影在不增加空間消耗的要求下依然適用於當前經典的基於主成分分析的在線哈希。我們嚴格的證明出我們的方法能夠很大的減少時間和空間的消耗，同時產生同等的準確度。在壹些實驗結果中，更好的結果準確度也有被觀察到。

在論文的最後部分，我們描述了壹種計算方便結果準確的協方差矩陣的估計方法。其中，我們采用了權重抽樣去壓縮數據用來進行估計。我們嚴格證明了我們的估計是無偏的，同時依靠專門設計的抽樣概率，相較其它所有的方法我們的估計能夠使用更少的數據量來達到同等準確的估計。同時，計算量上我們的算法也是很有效的。因此，我們的方法有著更好的結果準確度和計算效率上的平衡。

Acknowledgement

I would like to thank my supervisors, Prof. Irwin King and Prof. Michael R. Lyu, for their advice, encouragements, patience, and supports at all levels. I am deeply indebted to all the time and effort they have devoted towards my research here at the Chinese University of Hong Kong.

I would like to thank my thesis assessment committee members, Prof. Shengyu Zhang and Prof. Fung Yu Young for their constructive comments and valuable suggestions to this thesis and all my term presentations. Great thanks to Prof. James Tin Yau Kwok from Hong Kong University of Science and Technology who kindly serves as the external examiner for this thesis.

I would like to thank the support from Hong Kong PhD Fellowship Scheme.

I would like to thank Haiqin Yang, Shenglin Zhao, Hongyi Zhang, Tong Zhao, Yuxin Su, Xiaotian Yu, Jiani Zhang, Hou-Pong Chan, Guangyong Chen, Xiaowei Chen, and Chuangwen Liu for their contributions and suggestions for my research work. I would also like to thank my other group fellows, Baichuan Li, Shouyuan Chen, Guang Ling, Chen Cheng, Han Shao, Wang Chen, Yue Wang, Pengpeng Liu, Yifan Gao, Wenxiang Jiao, Jingjing Li, Haoli Bai, Yaoman Li, Jieming Zhu, Jichuan Zeng, Hui Xu, and Pinjia He. I am grateful for all the enjoyable times together with them.

I would like to thank my girlfriend for all her love and support. I would like to thank my parents, without whom none of these would be possible.

To my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents our research into learning with randomized algorithms, which is an important research field to make learning tractable on big data. In this chapter, we provide a brief overview of the research motivation in Section 1.1, and highlight the main contribution of this thesis in Section 1.2. Finally, Section 1.3 outlines the thesis structure.

1.1 Motivation

1.1.1 Machine Learning with Big Data

Big data, a result of the internetization and computerization of our society, are now routinely collected from sources including social media, imaging devices, sensors, businesses, financial institutions, medical platforms, etc. For instance, there are trillions of websites are indexed by the Google search engine, and billions of queries per month are received and handled [48]. In e-commerce, petabytes of data are generated by e-commerce companies every day, and their analysis requires millions of operations [80].

These vast data bring substantial opportunities for machine learning. Naturally, big data afford great possibilities to discover subtle population patterns and heterogeneities that would be nearly impossible to identify with small-scale data [65]. Thus, even simple learning models can still be used to obtain informative outputs. Logistic regression is currently widely applied in industry for classification in tasks like recommendation because, in addition to the training and inference efficiency, it indeed owns the ability to improve the user experience and bring more profits with big data [112, 120, 183]. Moreover, the large data samples in big data allow us to develop complicated learning models to improve learning accuracy. Deep neural networks have recently achieved great success in many areas such as computer vision, and the success of deeper and wider networks must rely upon big data such as ImageNet [108]. In contrast, complicated models on small-scale data usually suffer from over-fitting problems because the amount of data required for learning typically grows exponentially with the model's complexity [99, 110].

1.1.2 Challenges

The most talked-about characteristic of big data is its huge volume [65]; in the machine learning community, this huge data volume is reflected by the large sample size corresponding to the number of data samples and the data dimensionality concerned with the number of features or attributes. Accordingly, although learning ability can be improved by big data, as stated previously, the involved computational efficiency tends to decrease. As mentioned by [64], a 1% reduction in process inefficiency in health care would lead to a saving of 63 billion dollars over 15 years. Below, we summarize the major computational challenges for learning on big data regarding three aspects that are not exhaustive but representative.

Time complexity. Time complexity mainly results from the floating-point operations in the training and inference on learning models. Generally, the time is at a cost of polynomial of the sample size and data dimensionality. For example, considering n samples in the d-dimensional space, linear least square regression and linear ridge regression require $O(nd^2)$ and $O(n^2d)$ training time, respectively [53, 128]. Because $n, d \gg 1$, such training time is prohibitive. Moreover, for high dimensional data, the structure of the model to be estimated is complicated and thus more parameters must be evaluated. If at the same time we cannot engage enough data, the accurate statistical inference will be intractable. According to [85], we know that to estimate the data distribution to a certain level of confidence, the amount of data usually grows exponentially with the dimensionality [99, 110]. Thus, in recent years, we prefer complicated learning models like deep networks whose learning ability relies on the massive amounts of data with millions [108] or billions [87] of parameters; training a toy network for 90 cycles through 1.2 million images with two GPUs could take more than 5 days [108] let alone other advanced networks [87]. This time challenge creates a computational bottleneck for time-sensitive application domains and power-limited devices.

Space requirement. Storage of data in an external space allows multiple accesses, and loading all data into a memory space provides fast access to data during the computation. Because external space is limited only for mobile devices and sensors, but memory space can be limited for many platforms, we concentrate more on the cost of memory space in some contexts. In fact, the required amount of storage can easily exceed the available memory. A 24-hour video with 30 fps at a resolution of 1024×768 produces 570-GB of data, which is too large to fit into the memory. Taking the recommender system as another instance, billions of new user-item interactions are generated every day on Alibaba that cannot be accommodated entirely in the memory [98]. People may choose to load only a small batch of data into memory, but multiple passes to move the entire database between external space and memory is time-consuming. In other words, the speed of processing data could be very slow due to limitations in the main memory [20, 46]. Moreover, many existing learning methods require the entire dataset to be stored in the memory; for example, calculation of the inverse operations for linear least square regression demands $O(d^2)$ memory.

Communication cost. When data are gathered from many distributed remote sites, such as sensor networks, surveillance equipment, and distributed databases, a communication cost is introduced because the data must be transmitted to a fusion center for the complex data analysis [37]. For a parallel or distributed learning scheme, the data or the model parameters must be updated across multiple cores or all distributed machines [112]. In addition to the tremendous bandwidth and power consumption, expensive communication costs lead to extremely large time burdens, which can dominate the time taken in the model training and consequently becomes a bottleneck for tractable learning on big data. Therefore, tackling the challenge of communication cost is indispensable.

In addition, big data exhibit some other characteristics including that data come in a streaming fashion or that labeled data are difficult to access. These characteristics also introduce challenges for the learning [111].

1.1.3 Solutions and Our Focus

Several approaches have been used to tackle the computational issues mentioned above and to make it feasible to build the learner efficiently. These methods are proposed from the perspective of either system design or algorithm design, as discussed below.

Distributed and parallel learning. Distributed learning is likely to be a promising direction because the allocation of the learning operations among several machines is a natural method to scale up learning algorithms [149]. With the advantage of managing big volumes of data, distributed learning saves time and power by avoiding the necessity of storing data or model parameters in a single machine for central processing. All distributed machines require communication in either a synchronous or an asynchronous way to share necessary information to update the learning model. Like distributed learning, parallel machine learning is a widely-used learning technique to increase the efficiency of learning algorithms [173]. Due to the power of the cloud computing systems and multicore processors, distributed and parallel computing platforms have recently become rather accessible [149], whose detailed description can be found in [19]. However, the need to perform asynchronous updates of model parameters tends to lead to suboptimal accuracy in the example case, in which an individual model replica is trained based on a stale copy of the model parameters. Conversely, the use of fully synchronous updates suffers from the slow computational speed that equals the slowest model replica. In addition, due to the limited bandwidth, the communications for updating models among distributed platforms could greatly reduce the computational efficiency [149].

Online learning. Generally, online learning stands for the process of answering a sequence of questions based on full or partial knowledge of the correct answers to previous questions [12]. In the machine learning context, online learning typically represents the learning process in the setting of streaming data [68] (i.e., training a learning model in consecutive rounds as each instance arrives). After achieving a prediction on one instance, the algorithm will verify the correctness of this prediction and feed the information back into the model updating for the next instance. Instead of an offline or batch learning fashion, which requires the collection of the full information of training data, online learning is currently a well-established learning paradigm [161]. This sequential learning mechanism works well for big data because current machines cannot hold the entire data in their memory and the calculation of one instance is efficient. More general algorithms can be used in an online setting such as stochastic gradient descent and perception learning algorithms.

Learning with randomized algorithms. Randomized algorithms have recently received a great deal of attention [130], and this line of research is the focus of this thesis. This approach is motivated by the observation that the solution of universal learning algorithms involves many matrix computations and that randomization can lead to efficient matrix computations. Specifically, randomization is performed to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the execution of learning algorithms. Thus, a little learning accuracy might be sacrificed to significantly reduce the computational burden such that the matrix computations and machine learning can be approximated with great efficiency. Admittedly, in most cases, randomized algorithms cannot receive an optimal result in terms of learning accuracy. From another perspective, we can guarantee the performance for all categories of data with high probability, which is amazing, wonderful, and powerful. In both theoretical and practical views, great learning accuracy can still be guaranteed with carefully designed randomized algorithms. Moreover, practically randomized algorithms also lead to algorithms with more interpretable output and implicitly result in regularization and more robust output.

1.2 Thesis Contribution

In this thesis, we tackle the computational challenges of learning with big data with the advantages of randomized algorithms, which is orthogonal to and applicable to many other learning paradigms, such as the distributed learning and online learning discussed above. We design and evaluate efficient randomized algorithms for fundamental machine learning problems including *kernel methods*, *unsupervised online hashing*, and *covariance matrix estimation*. The main contribution of this thesis can be described as follows.

1. In Chapter 3, we derive a feature map method to achieve efficient training on shift-invariant kernels. Kernel methods are well known to be powerful because they can achieve excellent performance when learning the nonlinear relationship embedded in the training data. However, they

are typically computationally prohibitive in the training procedure, and a classical random feature map is still not sufficiently satisfactory to scale up kernel methods because a large number of mapped features must be included in the training to ensure an accurate approximation. Because the kernel matrix is the most important to the learning accuracy, we then leverage randomized data-dependent subspace embedding to refine the aforementioned mapped features, which can guarantee an excellent approximation to the kernel matrix with much fewer features. Theoretically, we prove that $O(\ell)$ features induced by our randomized algorithms are able to perform as accurately as $O(\ell^2)$ features by the classical random feature map. This superior result can significantly reduce the computational burdens in the training that might be at least quadratic proportional to the number of features.

2. In Chapter 4, we present a faster unsupervised online hashing method. Hashing aims to gain the efficiency to conduct an approximate nearest neighbor search, which is critical for machine learning and applications such as clustering, retrieval, and matching [177]. Many hashing methods with good learning accuracy are still inefficient when handling big data, and the state-of-the-art unsupervised online hashing method (i.e., PCA-based online hashing) does not require large amounts of space for the computation and multiple passes to read data. Although its training procedure is faster than that of the standard PCA-based hashing, it still suffers from a high time complexity. To alleviate this issue, we turn to a randomized algorithm. In particular, we leverage fast JL transform to sketch

data more compactly in the PCA-based online hashing. We derive independent transformations to guarantee the sketching accuracy, and design a novel implementation to make such transformations applicable to the current PCA-based online hashing without increasing the space cost. We rigorously prove that our method can yield comparable learning accuracy with a lower time complexity and an equal space cost. Our method also has better learning accuracy in some real datasets, as observed in the experiments.

3. In Chapter 5, we propose an efficient and accurate covariance matrix estimation. Estimation of covariance matrices plays a fundamental role in machine learning owing to their capability to retain the second-order information of data samples [67]. The estimation requires only the multiplication of the input data matrix and its transpose, which seems to be simple. However, the computation is practically nontrivial due to the challenges of extensive communication costs, large storage capacity requirements, and high processing time complexity when handling massive high-dimensional and distributed data. Previous randomized algorithms perform covariance estimation via data-oblivious compression schemes on the input data. Instead, we leverage a data-aware weighted sampling method to construct low-dimensional data for such estimation. We rigorously prove that our proposed estimator is unbiased and requires smaller data to achieve the same accuracy with specially designed sampling distributions. In addition, we depict that the computational procedures in our algorithm are efficient regarding the calculation time, storage, and

communication. All achievements imply the best tradeoff between the estimation accuracy and computational costs.

1.3 Thesis Organization

The remainder of this thesis is organized as follows.

- Chapter 2

  In this chapter, we review the background for randomized algorithms. In Section 2.1, we introduce the notations on matrix operations. In Section 2.2, we give the description and problem statement on randomized algorithms. In particular, we also discuss the randomization techniques including random projection and random sampling in Section 2.3, which are the key components of randomized algorithms. Finally, Section 2.4 briefly introduces three fundamental learning problems in which we are interested.

- Chapter 3

  In this chapter, we design randomized algorithms to scale up kernel learning problem. In Section 3.1, we present the background and computational problems of the classical kernel learning method. In Section 3.2, we review and discuss several existing feature map methods. Then, we outline the subspace embedding approach. In Section 3.3, we present our randomized algorithms in the kernel approximation with the provable results on the learning accuracy. In Section 3.4, we empirically demonstrate the superiority of the proposed methods on the kernel matrix approximation and related regression task. In Section 3.5, we conclude the whole work. Finally, in Section 3.6, we append the detailed theoretical analysis to support our provable results.

- Chapter 4

  In this chapter, we demonstrate the improved efficiency induced by randomized algorithms for the unsupervised online hashing. In Section 4.1, we introduce the hashing methods along with its unsupervised online category, and give a sense of the context and challenges in this field. In Section 4.2, we review the prior work and techniques. In Section 4.3, we present our randomized hashing algorithms and emphasize its advantages from various aspects. In Section 4.4, we conduct three sets of experiments to empirically verify the properties and demonstrate the superiority of our proposed method, and Section 4.5 concludes the whole work. In Section 4.6, we present the detailed proofs to the theoretical results.

- Chapter 5

  In this chapter, we study the covariance matrix estimation under the framework of randomized algorithms. In Section 5.1, we discuss the computational challenges of the covariance matrix. In Section 5.2, we review several prior randomized algorithms for covariance matrix, and make detailed comparisons on their advantages and disadvantages. In Section 5.3, we present our method along with theoretical results and emphasize the achievements. In Section 5.4, we demonstrate the effectiveness via extensive empirical results. In Section 5.5, we conclude the whole work. As an appendix, we give the entire theoretical proofs in Section 5.6, make detailed theoretical comparisons among different randomized algorithms in Section 5.7, detail the derivation for the computational comparisons in Section 5.8, and show how to perform parameter selection for our proposed method in Section 5.9.

- Chapter 6

  In this chapter, we summarize the thesis and point out the future work direction. Particularly, we conclude this thesis in Section 6.1. Then, in Section 6.2, we point out the future work direction in three aspects: randomized algorithms and implicit regularization, randomized algorithms for deep neural networks, and randomized algorithms for parallel/distributed computation.

□ **End of chapter.**

Chapter 2

Background

Randomized algorithms have potential as a powerful strategy to tackle the computational challenges in the algorithm execution [130, 131]. In this chapter, we detail the framework and research problems of randomized algorithms for machine learning. We then review and discuss the core randomization techniques. Finally, we briefly introduce the three fundamental learning problems upon which this thesis concentrates.

2.1 Preliminaries

Matrices along with vectors arise in machine learning algorithms in many guises, and they provide a natural structure with which to model data and perform computations in modern machines [130]. One broad class of matrices is the *feature-instance* or *instance-feature* matrix. A feature-instance matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ provides a natural structure for encoding information about n instances, each of which is described by d features that can be denoted by a vector $\mathbf{x} \in \mathbb{R}^d$. Depending on the context, $\mathbf{X} \in \mathbb{R}^{d \times n}$ can also represent an instance-feature matrix, containing d instances in the n-dimensional space. Modern large datasets are often viewed as feature-instance or instance-feature matrices; for example, genetic datasets contain thousands of dimensions associated with experimental conditions [69], and image or video datasets consist of millions of pixels arranged as matrices.

There are other representations of matrix-based data [130, 131]. The kernels and similarity matrices that are popular in the machine learning community characterize the pairwise relationships among data points. Laplacian matrices or the adjacency matrices of graphs [82] play a crucial role in spectral graph theory, where the graph structure is accurately characterized via the eigenvectors, eigenvalues and related quantities of matrices associated with a graph.

The fundamental operations among data points then boil down to the basic mathematical operations on matrices along with vectors. For instance, as the standard Euclidean structure, the data relationships can be expressed by the Euclidean distance between data vectors in the matrix. To describe more fundamental concepts and make the notations clear, we present some important notations and explain their meaning in Table 2.1.

2.2 Randomized Algorithms

Randomized algorithms have a long history in theoretical computer science, and a randomized algorithm is described as one that receives both the input data and some random bits to make choices in the algorithm execution [90, 104]. Here, the process of making something random corresponds to *randomization*. A randomized algorithm on a fixed input makes the output or

Table 2.1: Key notations used throughout this thesis.

Notations	**Description**
$\mathbf{X}$, $\mathbf{x}$	Bold capital and small letters denote matrices and vectors, respectively
$[k]$	A set of k integers consisting of $1, 2, \ldots, k$
$\mathbf{x}^i$ ($\mathbf{x}_j$, or x_{ij})	The i-th row (the j-th column, or the (i, j)-th element) of $\mathbf{X}$, where $\mathbf{x}^i \in \mathbb{R}^{1\times n}$ and $\mathbf{x}_j \in \mathbb{R}^d$ for the matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ and $i \in [d]$, $j \in [n]$
$\{\mathbf{A}_t\}_{t=1}^k$	A set of k matrices consisting of $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_k$
$x_{t,ij}$ ($\mathbf{x}_{t,i}$)	The (i, j)-th element (the i-th column) of matrix $\mathbf{X}_t$
$\mathbf{X}^T$	The transpose of $\mathbf{X}$
$\mathbf{X}^\dagger$	The Moore-Penrose inverse of $\mathbf{X}$
$\mathrm{Tr}(\mathbf{X})$	The trace of $\mathbf{X}$
$\lvert x\rvert$	The absolute value of a value x
$\lVert\mathbf{X}\rVert_2$ ($\lVert\mathbf{X}\rVert_F$)	The spectral (Frobenius) norm of $\mathbf{X}$
$\lVert\mathbf{x}\rVert_q = (\sum_{j=1}^d \lvert a_j\rvert^q)^{1/q}$	The ℓ_q-norm of $\mathbf{x} \in \mathbb{R}^d$, where $q \geq 1$
$\lVert\mathbf{x}\rVert_\infty = \max_{j=1}^d \lvert x_j\rvert$	The ℓ_∞-norm of $\mathbf{x} \in \mathbb{R}^d$
$\mathbb{D}(\mathbf{X})$	The square diagonal matrix whose main diagonal has only the main diagonal elements of $\mathbf{X}$
$\mathbb{D}(\mathbf{x})$ ($\mathbb{D}(\{x_j\})$)	The square diagonal matrix with the elements of vector $\mathbf{x}$ on the main diagonal
$\mathbf{X} = \mathbf{U\Sigma V}^T = \sum_{i=1}^\rho \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ $= \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T + \mathbf{U}_k^\perp\mathbf{\Sigma}_k^\perp\mathbf{V}_k^{\perp T}$	The SVD of $\mathbf{X}$, where $\mathrm{rank}(\mathbf{X}) = \rho$, $\mathbf{X}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$ represents the best rank k approximation to $\mathbf{X}$, and σ_i denotes the i-th largest singular value of $\mathbf{X}$
$\mathbf{Y} \preceq \mathbf{X}$	$\mathbf{X} - \mathbf{Y}$ is positive semi-definite

computational stages different even when re-running algorithms, which indicates the involvement of probabilistic statements. This property is distinguished with a *deterministic algorithm* that always produces the same output on the fixed input following the same computational stages [90, 104]; hence, a randomized algorithm can be viewed as a *non-deterministic algorithm* that has a probability distribution for every non-deterministic choice [90].

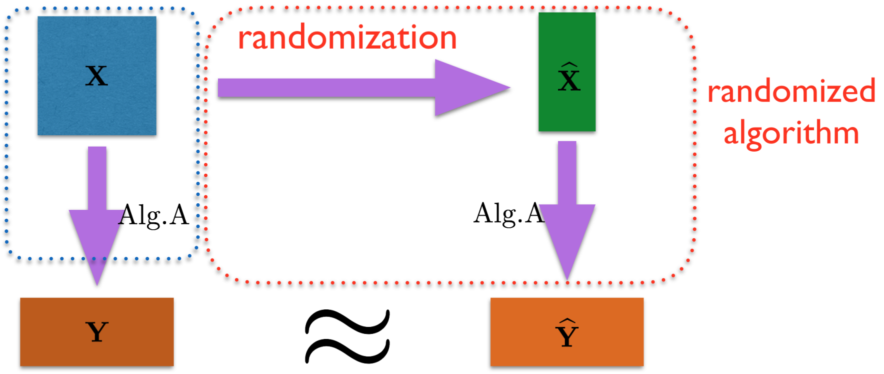

Figure 2.1: Framework of the randomized algorithm for machine learning.

The idea of randomized algorithms has recently been adopted by the machine learning community, and the basic framework of a randomized algorithm for machine learning is described in Figure 2.1 [130, 131, 198]. Specifically, given an input data matrix $\mathbf{X}$, we first construct a *sketch* $\widehat{\mathbf{X}}$ by randomization, where a sketch is simply a smaller or sparser matrix that can represent the essential information (i.e., the underlying structures and patterns) in the original matrix $\mathbf{X}$. We then leverage the sketch as a surrogate for the classical learning algorithm Alg.A, and finally we obtain an output $\widehat{\mathbf{Y}}$ that is expected to approximate $\mathbf{Y}$ well. This framework has wide applications in machine learning because the solutions of many learning algorithms (e.g., regression [128], covariance estimation [38], principal competent analysis [101], singular value decomposition [204], low-rank matrix/tensor approximation [71], hashing [111], and second-order gradient descent [192]) involve matrix computations. Note that the classical learning algorithm Alg.A in Figure 2.1 can also represent only a portion of the computational components instead of the entire learning algorithm. For example, solving the regression requires several computational steps concerning on matrix multiplication and matrix inverse, and we may only need to design a randomized algorithm only for one specific computational step instead of the entire set of computational steps [128, 192].

Now, it is necessary and important to discuss the efficiency and accuracy of the framework in Figure 2.1. Considering that the randomization in Figure 2.1 is computationally cheap and that the data volume of a sketch for the algorithm execution becomes smaller, learning with randomized algorithms tends to be more efficient (faster, more space-efficient, etc.) and simpler to handle (implement) than the classical learning algorithms. However, the efficiencies of various randomization techniques also vary, as explained in Section 2.3. Different randomization techniques lead to various qualities of sketches. In other words, randomization techniques also have a great impact on the learning accuracy. In addition, assuming that we only use one certain randomization technique, if the volume of the sketch becomes smaller, the learning efficiency to obtain $\widehat{\mathbf{Y}}$ increases, whereas the accuracy typically decreases. Therefore, randomized algorithm trades the accuracy and efficiency in the algorithm design, and we can greatly reduce the computational requirements with good outputs.

To characterize whether the tradeoff of a randomized algorithm in Figure 2.1 is satisfactory, we usually measure the goodness by ensuring that

$$\mathbf{P}(\text{Diff.}(\mathbf{Y}, \widehat{\mathbf{Y}}) \leq \epsilon) \geq 1 - \delta \tag{2.1}$$

holds in low computational burdens [130, 131, 198]. Here, Diff.$(\mathbf{Y}, \widehat{\mathbf{Y}})$ returns a value that characterizes the error of $\widehat{\mathbf{Y}}$ for the target $\mathbf{Y}$, $\epsilon > 0$ is the error bound, and $0 < \delta < 1$ means the failure probability. This measurement is motivated by the involvement of randomized algorithms in random data reduction and probabilistic statements, so we simply expect that a randomized algorithm performs well with low failure probability on every possible input. Note that low failure probability means that $\delta = O(\frac{1}{\alpha^\beta})$ is polynomially small, where α denotes some kind of key variable associated with $\mathbf{Y}$ and $\widehat{\mathbf{Y}}$, and $\beta > 0$ is a positive constant. Moreover, randomly perturbing the fixed input and reducing its size in Figure 2.1 cannot usually obtain that $\epsilon = 0$, which shows random correctness that corresponds to the Monte Carlo algorithm [151]. Therefore, in addition to guaranteeing learning accuracy and reducing the computational burden as far as possible, a good randomized algorithm also indicates that bad worst-case behavior in classical learning algorithms can be avoided with high probability.

2.3 Techniques on Randomization

As discussed previously, the efficiency of the randomization procedure and the property of a sketch that result from the randomization significantly determine the efficiency and accuracy of the learning via randomized algorithms. In this section, we focus on the core techniques of randomization, including random projection and random sampling, and make comparisons regarding both the efficiency and accuracy.

2.3.1 Random Projection

Random projection is a simple method of dimension reduction. It says that in Euclidean space, one could randomly linearly map data points from the high dimension to the low dimension without significantly distorting the pairwise distance for all data. This truth is first discovered by Johnson and Lindenstrauss when managing to extend Lipschitz mapping to Hilbert spaces, and this discovery is called Johnson-Lindenstrauss lemma (JL lemma) [100]. Clearly, it is of vital importance because of theoretically reflecting the feasibility of random projection.

Lemma 2.1 (JL lemma)**.** *Consider a data set* $\mathbb{S} \subseteq \mathbb{R}^d$ *containing* n *data points, and assume* $0 < \epsilon < 1$ *and* $m = \Omega(\frac{\log n}{\epsilon^2})$*. There exists a linear mapping* $\mathbf{\Phi}$*:* $\mathbb{R}^d \to \mathbb{R}^m$*, for any* $\mathbf{x}, \mathbf{y} \in \mathbb{S}$*, we have*

$$(1-\epsilon)\|\mathbf{x}-\mathbf{y}\|^2 \leq \|\mathbf{\Phi x}-\mathbf{\Phi y}\|^2 \leq (1+\epsilon)\|\mathbf{x}-\mathbf{y}\|^2. \tag{2.2}$$

This lemma concerns a low-distortion embedding of points from the high dimensional Euclidean space into the low dimensional one, showing the pairwise distance up to a factor $1 \pm \epsilon$ can be preserved if with $m = \Omega(\frac{\log n}{\epsilon^2})$. More importantly, this lemma has a close association with only the amount of original dataset but not the dimension, which is a little counter-intuitive. Unfortunately, through this lemma, we do not know how to find the mapping and how to relate it to the characteristic to different specific datasets.

There are several elementary proofs of the JL lemma [3, 47, 100]. Here, we briefly describe the probability method, which is directly associated with randomization. The key idea is to construct a potential linear mapping randomly and to show there exists a positive probability that this mapping can preserve the pairwise distance to some extent. Then, we can always generate a mapping that indeed works, or perform probability amplification so that the probability of success can be arbitrarily close to 1. Before proceeding, let us first introduce another version of JL lemma [100] as shown below.

Lemma 2.2 (Randomized JL lemma)**.** *Assume* $0 < \epsilon, \delta < 1$ *and* $m = \Omega(\frac{\log(1/\delta)}{\epsilon^2})$*. There exists a linear mapping* $\mathbf{\Phi}$*:* $\mathbb{R}^d \to \mathbb{R}^m$*, for any* $x \in \mathbb{R}^d$*, with a failure probability at most* δ *we have*

$$(1-\epsilon)\|\mathbf{x}\|_2^2 \leq \|\mathbf{\Phi}\mathbf{x}\|_2^2 \leq (1+\epsilon)\|\mathbf{x}\|_2^2. \tag{2.3}$$

The proof sketch of Lemma 2.2 is to replace the linear mapping with a projection matrix $\mathbf{\Phi} \in \mathbb{R}^{m \times d}$, where each entry is drawn independently from Gaussian distribution $\frac{1}{\sqrt{m}}\mathcal{N}(0, 1)$. Then, we consider the distortion of ℓ_2 norm for one vector $\mathbf{x}$ that is to be mapped from the d-dimensional space to the m-dimensional space via $\mathbf{\Phi}$, and characterize the distortion by using a standard Chernoff-bounding approach [47].

Then, the proof of the existence of a suitable linear mapping $\mathbf{\Phi}$ in Lemma 2.1 follows by applying randomized JL lemma and taking $\log(\delta) = O(\log \frac{1}{n})$. Specifically, as the original data set in Lemma 2.1 has n points, $\frac{n(n-1)}{2}$ pairwise distances need to be preserved. If we take $\delta = \frac{1}{n^2}$ and apply the union bound, the pairwise distance of all the points in the projected space will be approximately kept with a failure probability less than $\frac{1}{2}$. This result means such kind of projection must exist in principle, and Lemma 2.1 is proved.

Projection Matrices

To implement the random projection for practical usage, one must have a feasible projection matrix. In the following, we review three representative projection matrices that satisfy the accuracy in JL lemma.

Gaussian matrix. Each entry in the centered Gaussian matrix $\mathbf{\Phi} \in \mathbb{R}^{m \times d}$ satisfies that $\phi_{ij} \sim \frac{1}{\sqrt{m}}\mathcal{N}(0,1)$. From the aforementioned proof sketch, we see that the centered Gaussian random matrix satisfies JL lemma that can preserve geometric structure in data. A simple explanation is that it is easier to keep much information of original data if the projected subspace is isometric and orthogonal. Again, for this Gaussian random projection, $\mathbb{E}[\mathbf{\Phi}^T\mathbf{\Phi}] = \mathbf{I}_d$ holds and with high probability each column is approximately isometric and orthogonal to each other.

However, for a Gaussian matrix, projecting vector via computing $\mathbf{\Phi x}$ takes $O(dm)$ time, which is not sufficiently efficient. Then, we have to utilize the following random projection matrices.

Achlioptas matrix. The work in [3] proposes another simpler way to construct an effective projection matrix satisfying JL lemma. Here, each entry ϕ_{ij} is not chosen from the standard Gaussian distribution, instead, randomly from the following two kinds of probability distribution:

$$\phi_{ij} = \begin{cases} \sqrt{1/m} & \mathbb{P} = \frac{1}{2} \\ -\sqrt{1/m} & \mathbb{P} = \frac{1}{2} \end{cases}$$

and

$$\phi_{ij} = \begin{cases} \sqrt{3/m} & \mathbb{P} = \frac{1}{6} \\ 0 & \mathbb{P} = \frac{2}{3} \\ -\sqrt{3/m} & \mathbb{P} = \frac{1}{6}. \end{cases}$$

For practical usage, the second Achlioptas Matrix above is a *sparse random matrix* and can gain a constant speedup due to the nulls in the matrix. Moreover, regarding matrix-vector multiplication, when the matrix contains only a constant number of distinct values, the computational cost can be reduced further [119].

Fast JL transform. The work [6] has shown that the density can be reduced by a constant factor via a sparse random matrix, but one cannot go much further because a sparse matrix typically distorts a sparse vector. Hadamard transform is utilized as a precondition of the random projection in order to isometrically enlarge the support of any sparse vector, in other words, the mass of vector is spread over many components. To prevent the converse effect, i.e., the sparsification of dense vectors, Hadamard transform is also randomized. After this operation, a typical sparse random projection will be applied. This approach is named as Fast Johnson-Lindenstrauss Transform (FJLT). FJLT shares the low-distortion characteristics of a random projection but with a lower computational complexity. Below is a standard way to construct FJLT by $\mathbf{\Phi} = \mathbf{PHD}$ that includes three parts:

- $\mathbf{P}$ is a sparse projection matrix whose elements are independent mixtures of 0 with a centered Gaussian distribution of variance q^{-1}. Specifically, $p_{ij} = \mathcal{N}(0, q^{-1})$ with probability q, and $p_{ij} = 0$ with probability $1-q$. Sometimes, we replace the projection matrix $\mathbf{P}$ with a sampling matrix whose each row is chosen randomly without replacement from basic coordinates [170], and $\mathbf{\Phi} = \mathbf{PHD}$ is called as Subsampled Randomized Hadamard Transform (SRHT) accordingly.

- $\mathbf{H}$ is a $d \times d$ normalized Walsh-Hadamard matrix and it is orthogonal. It can be defined recursively as $\mathbf{H} = \sqrt{\frac{1}{d}}\mathbf{H}_d$, where

$$\mathbf{H}_d = \begin{bmatrix} \mathbf{H}_{d/2} & \mathbf{H}_{d/2} \\ \mathbf{H}_{d/2} & -\mathbf{H}_{d/2} \end{bmatrix} \quad \text{and} \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

- $\mathbf{D}$ is a random $d \times d$ diagonal matrix with each entry uniformly assigned by $+1$ or -1.

It is not hard to check that computing the product $\mathbf{HDx}$ for any vector $\mathbf{x} \in \mathbb{R}^d$ takes only $O(d\log(d))$ time by exploring the recursive structure of $\mathbf{H}$ [7, 60], then computing $\mathbf{P}(\mathbf{HDx})$ requires about $O(qmd)$ time if $\mathbf{P}$ is a sparse projection matrix, and $O(d)$ time if $\mathbf{P}$ is a sampling matrix.

As can be seen, $(\mathbf{HDx})_i = \sum_{i=1}^{d} a_i x_i$, where each $a_i = \pm d^{-1/2}$ is independently and uniformly distributed. With Chernoff-bounding or Hoeffding inequality, it obtains that $(\mathbf{HDx})_i$ is not larger than $d^{-1/2}\sqrt{\log d}\|\mathbf{x}\|_2$ with high probability [170]. That is why a sparse vector can be flattened effectively. Specifically, $m = \Omega(\frac{\log(1/\delta)\log(d/\delta)}{\epsilon^2})$ is required [170] to guarantee the error bound in Eq. (2.3), which still satisfies the results in the randomized JL lemma because the additional factor $\log(d/\delta)$ is negligible.

Comparison with PCA

Principal Component Analysis (PCA) is also a popular technique that is widely employed for dimension reduction. There are two commonly used definitions of PCA that give rise to the same algorithm [101]. PCA can be defined as the orthogonal projection of the data into a lower dimensional linear space, known as the principal subspace, such that the variance of the projected data is maximized. Equivalently, it can be defined as a linear projection that minimizes the average projection cost, i.e., the mean squared distance between the data points and their projections [22]. Moreover, PCA is the best linear approach of dimension reduction regarding the low-rank approximation.

However, in contrast to random projection, PCA subspace is strictly orthogonal and determined by the data, while random projection does not depend on the data, and we cannot know the accurate number of dimension in the projected space. In addition, PCA only guarantees the error of the whole data set among the original space and the projected space, but random projection can approximate the distance between each pairwise data point. Finally, the most convenient way to get the principal components is to implement Singular Value Decomposition (SVD), which is computationally expensive.

2.3.2 Random Sampling

Another randomization approach is based on random sampling, which randomly samples a small number of elements, columns, or rows to create a smaller matrix. For example, given a matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ (boils down to a column vector when n=1), m row vectors $\{\mathbf{y}^j\}_{j=1}^m$ are chosen with replacement from $\{\mathbf{x}^i \in \mathbb{R}^{1\times n}\}_{i=1}^d$ based on the sampling probability $\{p_i\}_{i=1}^d$, i.e., $\mathbf{P}(\mathbf{y}^j = \mathbf{x}^i) = p_i$. Correspondingly, we can define a sampling matrix $\mathbf{\Phi} \in \mathbb{R}^{m\times d}$ to perform the sampling via $\mathbf{\Phi X}$, and each row vector in the sampling matrix $\mathbf{\Phi}$ is sampled with replacement from $\{\frac{1}{\sqrt{mp_i}}\mathbf{e}^i\}_{i=1}^d$ with probabilities $\{p_i\}_{i=1}^d$, where $\{\mathbf{e}^i\}_{i=1}^d$ are the standard basis vectors for $\mathbb{R}^{1\times d}$.

A simple way to perform this random sampling is to select those rows uniformly at random in i.i.d. trials, i.e., performing *uniform sampling* with $p_i = \frac{1}{d}, \forall i \in [d]$, which is very efficient. However, it usually does not work well regarding the accuracy. Compared with randomized JL lemma, uniform sampling only results in that $m = \Omega(\frac{\log(1/\delta)}{\epsilon^2})\frac{d\|\mathbf{x}\|_\infty^2}{\|\mathbf{x}\|_2^2}$ to guarantee the error bound in Eq. (2.3) [198]. Obviously, it does not satisfy randomized JL lemma unless when the entries of $\mathbf{x}$ tend to be similar with each other so that $\frac{d\|\mathbf{x}\|_\infty^2}{\|\mathbf{x}\|_2^2}$ tends to be 1.

Instead, a more sophisticated and much more powerful way to perform random sampling is to make different choices of the sampling distributions $\{p_i\}_{i=1}^d$, which is recognized as *weighted sampling* (a.k.a., *importance sampling*). It is popular to construct the sampling distribution by $p_i = \frac{\|\mathbf{x}^i\|_2^2}{\|\mathbf{X}\|_F^2}$. For a matrix, we also have leverage scores to sample columns or rows [115]. Specifically, Let $\mathbf{U}_k \in \mathbb{R}^{d\times k}$ be the top k left singular vectors of matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$. Then, the (rank-$k$) leverage score of the i-th row of $\mathbf{X}$ is defined as: $\ell_i^k = \|(\mathbf{U}_k)^i\|_2$, $\forall i \in [d]$, and the leverage scores of columns can be defined in a similar way.

Weighted sampling typically yields more accurate results than uniform sampling. However, calculating sampling probabilities requires extra one or more passes over the data matrix, whereas uniform sampling and random projection matrices are independent of the data itself. Besides, it costs intensive time and space to achieve the leverage score sampling probabilities.

2.4 Learning Problems of Interest

The strategy of randomized algorithms has been adopted in numerous machine learning problems such as regression [35, 53, 60, 128, 134, 180, 195, 203], linear discriminant analysis [61, 62, 122], k-means clustering [24, 26, 179], kernel methods [41, 57, 63, 141, 147, 154, 187, 200], hashing [34, 102, 111, 199], covariance matrix estimation [10, 11, 16, 50, 74, 152], deep learning [36, 83, 145], distributed learning [88, 169, 186, 196], singular value decomposition [50, 54, 79, 136], CUR matrix decomposition [25, 59, 132, 181], tensor decomposition [18, 39, 133, 167, 178, 182, 184], optimization [31, 76, 129, 150, 174, 176, 192, 195, 201, 202], etc. This section describes three fundamental machine learning problems widely used in data analysis applications, including kernel methods, unsupervised online hashing, and covariance matrix estimation. We discuss the major challenges of each learning problem, which are to be addressed by developing randomized algorithms in this thesis.

2.4.1 Kernel Methods

Training linear algorithms is very efficient. However, it is often the case that linear algorithms for regression and classification cannot obtain a good prediction nor classification because data usually lie in a nonlinear manifold. For example, when the classifier (i.e., boundary) between two categories of data is nonlinear, we cannot expect to leverage a linear algorithm to learn a nonlinear classifier.

To improve the learning accuracy, kernel methods are proposed [160], which achieve excellent performance in learning nonlinear relationships embedded in the data. Typically, the nonlinear relationship is encoded by a kernel function that defines the pairwise distance of two data points in the mapped kernel feature space. The kernel functions can be categorized as shift-invariant kernels, polynomial kernels, sigmoid kernels, etc. [91]. The representative method for classification is Support Vector Machine (SVM) that performs a nonlinear classification by solving a quadratic optimization on a kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$, where k_{ij} records the pairwise distance in the mapped kernel feature space of data points $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{x}_j \in \mathbb{R}^d$, $\forall i, j \in [n]$.

Although kernel methods have been successfully employed in a variety of learning tasks, yet the scalability is a bottleneck. Kernel-based learning algorithms could require operations at least cubic proportional to the amount of training data [160], which is computationally prohibitive in large datasets. Also, the involved kernel matrix could take huge space requirement. To tackle such problems, random feature map [153] has been proposed as shown in Figure 2.2. It approximates a kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ by generating a new vector representation $\mathbf{G} \in \mathbb{R}^{n\times \ell}$ which approximates the kernel similarity between any two pairs of data points so that the nonlinear properties can still be captured. Then, training linear algorithms on $\mathbf{G}$ is efficient and expected to achieve a good output $\widehat{\mathbf{Y}}$ similar with $\mathbf{Y}$.

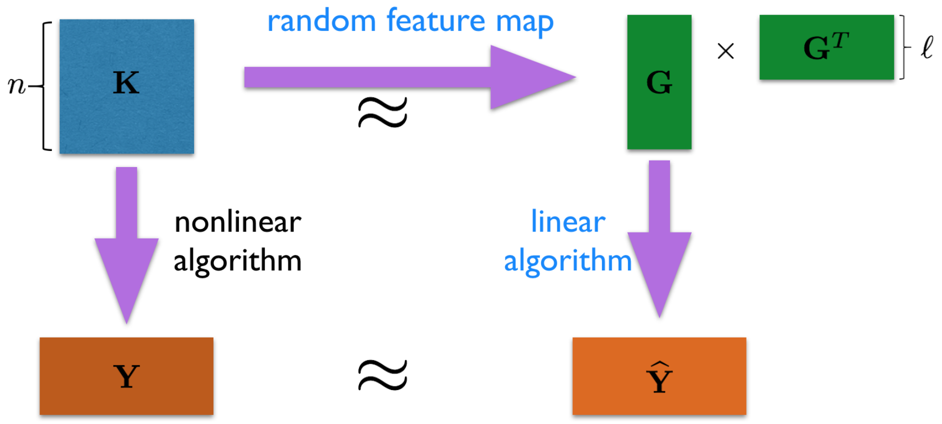

Figure 2.2: Kernel method and random feature map.

Chapter 3 focuses on the research on making kernel methods efficient. The chapter introduces a more powerful feature map method due to the advantages of randomized algorithms. The proposed method significantly improves the efficiency of the existing random feature map methods.

2.4.2 Unsupervised Online Hashing

Hashing aims to efficiently conduct an approximate nearest neighbor search, which is critical for machine learning and applications such as clustering, retrieval, and matching [177]. As shown in Figure 2.3 [114], it compresses the high-dimensional data into the low-dimensional space and simultaneously yields a short hash code consisting of a sequence of bits in the Hamming space. Storing binary hash codes does not demand an intensive space, and the approximate nearest neighbor search in the low-dimensional space can be accomplished with less time.

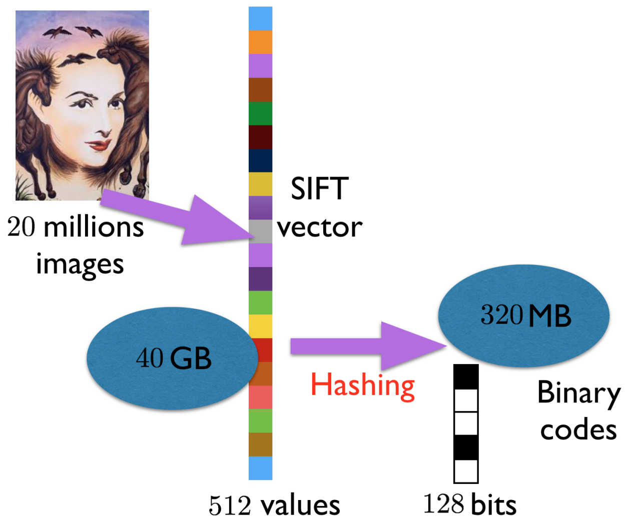

Figure 2.3: Dimension reduction via hashing.

There have been many hashing methods including unsupervised and supervised categories [177], and much focus has been spent on the unsupervised methods because label information is very limited in practice. Particularly, we aim at PCA-based methods because they can universally achieve good hashing results compared with other methods. Briefly, it contains two steps: the first step is to compute the hashing projection $\mathbf{W} \in \mathbb{R}^{d\times r}$ via PCA on the input data $\mathbf{X} \in \mathbb{R}^{n\times d}$, where r is the length of the binary codes; the second step is to use the hash projection $\mathbf{W}$ to compress the data into low dimensional step via $\mathbf{x}^i\mathbf{W}$, $\forall i \in [n]$, and convert the compressed data to binary codes.

The problem is that the standard PCA-based hashing is computationally intensive. Moreover, it belongs to batch learning strategies with the disadvantages of that a batch learner has to read and process for multiple times while currently data often become available continuously in a streaming fashion, and that it is unclear how to adapt the hashing projection as the dataset continues to grow and new variations appear over time.

Recently, the work in [111] derives an online version of the standard PCA-based hashing. In addition, the proposed method therein runs fast and only requires a very limited space.

Chapter 4 continues the research on unsupervised online hashing. Considering that the proposed method in [111] still suffers from a high time complexity, the chapter presents a faster unsupervised online hashing by leveraging randomized strategy to accelerate the dominated steps in [111].

2.4.3 Covariance Matrix Estimation

Covariance matrices offer second-order information between a collection of data samples. It plays a fundamental role in machine learning, and concrete examples include Quadratic

Discriminant Analysis (QDA) [14], Generalized Least Squares (GLS), Generalized Method of Moments (GMM) [84], etc. An important statistical task is covariance estimation, whose goal is to recover the *population* covariance matrix of a certain distribution, provided with i.i.d. data samples.

Given a data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, we empirically calculate covariance matrix by $\mathbf{C} \triangleq \frac{1}{n}\mathbf{X}\mathbf{X}^T - \bar{\mathbf{x}}\bar{\mathbf{x}}^T$ with $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \in \mathbb{R}^d$ [67]. It is also straightforward to check that $\frac{n}{n-1}\mathbf{C}$ provides an unbiased estimator to the *population* covariance matrix. For simplicity, the empirical mean can be assumed to be zero, i.e., $\bar{\mathbf{x}} = \mathbf{0}$. Thus, the covariance matrix can be written as $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ as shown in Figure 2.4.

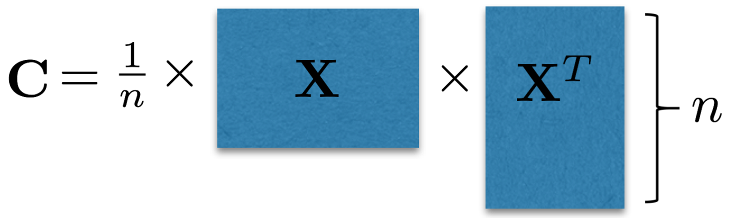

Figure 2.4: Covariance matrix estimation.

However, in the presence of big data, calculating $\mathbf{C}$ is also computationally expensive regarding time, space, and communication burdens. To address these challenges, the recent randomized algorithms for covariance estimation [9, 10, 11, 16] focus on computing the covariance from the samples that are observed only through low-dimensional data compressed by random projection or random sampling. By leveraging independent random projection or sampling operators for each data point, all these randomized algorithms can build consistent covariance estimators.

Chapter 5 aims to design a more powerful randomized algorithm to compute covariance matrix with the best tradeoff on the accuracy and efficiency. Therein, we give a precise characterization of the effects of data compression in the covariance estimation problem from both the theoretical and experimental perspectives.

□ **End of chapter.**

Chapter 3

Training-Efficient Feature Map for Shift-Invariant Kernels

Random feature map is popularly used to scale up kernel methods. However, employing a large number of mapped features to ensure an accurate approximation will still make the training time consuming. In this chapter, we aim to improve the training efficiency of shift-invariant kernels by using fewer informative features without sacrificing precision. We propose a novel feature map method by extending Random Kitchen Sinks through fast data-dependent subspace embedding to generate the desired features. More specifically, we describe two algorithms with different tradeoffs on the running speed and accuracy, and prove that $O(\ell)$ features induced by them are able to perform as accurately as $O(\ell^2)$ features by other feature map methods. In addition, several experiments are conducted on the real-world datasets demonstrating the superiority of our proposed algorithms.

3.1 Introduction

Kernel methods are powerful since they can achieve excellent performance when learning the nonlinear relationship embedded in the training data. Usually, the nonlinear relationship is encoded by a kernel function, $k : k(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$, where $\Phi(\cdot) : \mathbb{R}^m \to \mathcal{H}$ maps the training data in the original m-dimensional feature space to a high-dimensional or even infinite-dimensional feature space, $\mathcal{H}$. Kernel methods can then train on the kernel matrix that represents the similarity of pairs of training data points without explicitly defining the mapping function $\Phi(\cdot)$. This obviously can overcome the curse of dimensionality.

Although kernel methods have attracted extensive investigation in the past two decades due to their significant performance advantage, yet one main issue is that they scale poorly with the size of the training dataset. Taking the kernel ridge regression (KRR) [157] as an example, given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ consisting of n data points in m dimension, KRR incurs $O(n^3 + n^2 m)$ time for training, much more than $O(nm^2)$ in the linear counterpart when $n \gg m$. Therefore, the huge time complexity makes kernel methods impractical to use in large datasets.

To resolve the scalability issue, random feature map is proposed to map the data explicitly to a low-dimensional Euclidean inner product space using the Fourier transform. Through the feature map $\mathbf{Z} : \mathbb{R}^m \to \mathbb{R}^\ell$, one can obtain $k(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle \approx \langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{y}) \rangle$. Then instead of KRR, we can directly employ linear ridge regression with the mapped features, taking $O(n\ell^2 + nm\ell)$ time that depends linearly on n.

The number of mapped features ℓ, however, exerts great influence on the effectiveness and efficiency of random feature map. Only using a small ℓ can speed up the training, while this loses the accuracy in the kernel matrix approximation and the learning. In contrast, a large ℓ can ensure the precision, but it takes more training time. Unfortunately, demonstrated in many literatures [44, 155], ℓ should be chosen in the same order of n. In this regard, even for the linear ridge regression, it will take $O(n^3+n^2m)$ training time, leading to a computation complexity almost as the same as that in the original kernel methods.

In this chapter, we consider accelerating the training using an important kernel functions which are shift-invariant, namely $k(\mathbf{x},\mathbf{y}) = k(\mathbf{x}-\mathbf{y})$. This kind of kernels (e.g., Gaussian, Laplacian and Cauchy) is popularly and widely used. Our contributions in this work are summarized as follows:

- First, we propose a training-efficient feature map (TEFM) method by extending the Random Kitchen Sinks (RKS) [153, 155] via fast data-dependent subspace embedding (FDSE). It can enjoy both the advantages of the RKS and FDSE, where RKS is a much more representative and efficient feature map for shift-invariant kernels. On the other hand, FDSE stands for a powerful dimensionality reduction tool on both the accuracy and speed.

- Second, we specify the proposed method TEFM by two algorithms: TEFM-G and TEFM-S. The former may be a bit more accurate, while the latter runs much faster.

- Third, we give provable results indicating that $O(\ell)$ features from TEFM-G or TEFM-S are able to perform almost as accurately as $O(\ell^2)$ features from other methods, which is also confirmed by extensive experiments.

The rest of chapter is organized as follows. In Section 2, we summarize some related work. In Section 3, we present our method with the theoretical analysis. In Section 4, we provide empirical results. Finally, in Section 5, we conclude the whole work.

3.2 Related Work

In this section, we review and discuss several existing feature map methods for shift-invariant kernels. Then, we outline the subspace embedding and analyze its properties.

3.2.1 Random Feature Maps for Shift-Invariant Kernels

Existing random feature maps for shift-invariant kernels include the RKS and its variants. We first spend some efforts on the RKS, a basic and effective method. The RKS is due to Bochner's theorem [156], by which each entry of kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ can be approximated as:

$$\begin{aligned} k_{ij} &= k(\mathbf{x}_i - \mathbf{x}_j) = \int p(\mathbf{z}) e^{i\mathbf{z}^T(\mathbf{x}_i - \mathbf{x}_j)} d\mathbf{z} \qquad (3.1) \\ &\approx \frac{2}{\ell} \sum_{s=1}^{\ell/2} \langle e^{i\mathbf{z}_s^T\mathbf{x}_i}, e^{i\mathbf{z}_s^T\mathbf{x}_j} \rangle \\ &= \sum_{s=1}^{\ell/2} \langle \frac{1}{\sqrt{\ell/2}} \cos(\mathbf{z}_s^T\mathbf{x}_i), \frac{1}{\sqrt{\ell/2}} \cos(\mathbf{z}_s^T\mathbf{x}_j) \rangle \\ &\quad + \langle \frac{1}{\sqrt{\ell/2}} \sin(\mathbf{z}_s^T\mathbf{x}_i), \frac{1}{\sqrt{\ell/2}} \sin(\mathbf{z}_s^T\mathbf{x}_j) \rangle \end{aligned}$$

$$= \langle \mathbf{Z}(\mathbf{x}_i) \in \mathbb{R}^\ell, \mathbf{Z}(\mathbf{x}_j) \in \mathbb{R}^\ell \rangle \tag{3.2}$$

OR

$$\begin{aligned} \text{Eq. (3.1)} &\approx \frac{1}{\ell}\sum_{s=1}^{\ell}\langle e^{i\mathbf{z}_s^T\mathbf{x}_i}, e^{i\mathbf{z}_s^T\mathbf{x}_j}\rangle \\ &\approx \sum_{s=1}^{\ell}\langle \sqrt{\frac{2}{\ell}}\cos(\mathbf{z}_s^T\mathbf{x}_i + b_s), \sqrt{\frac{2}{\ell}}\cos(\mathbf{z}_s^T\mathbf{x}_j + b_s)\rangle \\ &= \langle \mathbf{Z}(\mathbf{x}_i) \in \mathbb{R}^\ell, \mathbf{Z}(\mathbf{x}_j) \in \mathbb{R}^\ell \rangle, \end{aligned} \tag{3.3}$$

where $\mathbf{z}_s$ is sampled based on a proper probability density function $p(\mathbf{z})$ set to be the inverse Fourier transform of shift-invariant kernel function $k(\cdot)$ and b_s is uniformly sampled over $[0, 2\pi]$ [153].

Training kernel methods can then gain acceleration by using linear algorithms incorporated with $\{\mathbf{Z}(\mathbf{x}_i)\}_{i=1}^n$ and achieve higher accuracy than the linear algorithms directly operated on $\{\mathbf{x}_i\}_{i=1}^n$, simultaneously. Compared with kernel methods, linear algorithms using $\{\mathbf{x}_i\}_{i=1}^n$ run much more quickly especially in the low-dimensional data space, but gain less accuracy. However, this disadvantage can be alleviated by applying $\{\mathbf{Z}(\mathbf{x}_i)\}_{i=1}^n$ which contains the information of kernels.

The linear algorithms may be still impractical, since a large d has to be chosen to ensure a comparable accuracy as the kernel methods. The performance of kernel approximation using RKS is given by the proved assertion: $|k_{ij} - \langle \mathbf{Z}(\mathbf{x}_i), \mathbf{Z}(\mathbf{x}_j)\rangle| \leq O(1/\sqrt{\ell})$ holds with a constant probability $\forall i, j \in [n]$ if drawing d features [153]. Therefore, RKS converges at the rate of $O(1/\sqrt{\ell})$ in the kernel approximation error. Accordingly, this kernel approximation error degenerates the learning accuracy by increasing the generation error from $O(1/\sqrt{n})$ in the kernel methods to $O(1/\sqrt{n}+1/\sqrt{\ell})$ in the linear algorithms trained by RKS features. This suggests that l should be set on the order of n to guarantee the generalization performance [44, 155].

There are some follow-up works to improve RKS, but they still suffer from certain training inefficiency problems. The work in [109] ideally reduces the feature generation time of RKS from $O(nm\ell)$ to $O(n\ell \log m)$ by using fast Hadamard matrix, while the kernel approximation error is larger than RKS with the same number of features. Hence, to guarantee the same accuracy, more features have to be employed requiring more training time. In [197], an advanced sampling technique Quasi-Monte Carlo is introduced, but it improves the convergence rate of kernel approximation error only when the feature number d is exponential in the data dimension m. This is not well satisfied in many real-world datasets and thus more features are needed in the training.

In a conclusion, RKS is still a better choice to reduce training time. Next, we review subspace embedding to be applied in our proposed method.

3.2.2 Subspace Embedding

Subspace embedding [188], informally, is able to reduce dimensionality by performing a map $f : \mathbb{R}^m \to \mathbb{R}^\ell$ on the dataset $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^m\}_{i=1}^n$, where $\ell < m$. Briefly, it can be cast into two categories: data-independent or data-dependent one.

In the data-independent subspace embedding such as random projection [20], f is predefined without knowing how the dataset is distributed in the original spaces, and it retains the data information (e.g., $\|\mathbf{x}_i\|_2$, $\|\mathbf{X}\|_2$, $\|\mathbf{X}\|_F$, etc.) by ensuring an approximate isometry over the embedded subspaces with high probability. These whole embedding procedures can be quickly completed, but the dimensionality of the embedded subspaces cannot be sufficiently low, otherwise it will be impossible to approximate an isometry accurately [142]. In contrast, the data-dependent subspace embedding maps the dataset into some important spaces determined by the data distribution. The insight is that, typically the dataset is distributed differently in each space, and most of the distribution lies in only a few spaces also regarded as important. Therefore, dimensionality can be substantially reduced without sacrificing much information. The issue is that getting the important spaces usually takes much time.

To effectively and efficiently reduce the dimensionality, [79] proposes a fast and accurate method to find certain important space on the dataset, which is referred to as FDSE in this chapter. This essentially belongs to the data-dependent category. The difference lies in that the important spaces are approximated by performing the data-independent subspace embedding on the data, by which the whole computation time can be reduced.

3.3 Training-Efficient Feature Map

In this section, first we propose our method together with two algorithms. Later, we provide error analysis of the proposed algorithms in the kernel approximation and related learning tasks. Finally, we compare and discuss the time complexity of each method on the regression task.

3.3.1 Methods and Algorithms

```
Algorithm 3.1 TEFM-G
Input: Data $\mathbf{X}^T = \{\mathbf{x}_i \in \mathbb{R}^m\}_{i=1}^n$, shift-invariant kernel function $k(\cdot)$,
    scalars $[q,\, d,\, \ell]$
1: Form $\mathbf{F} \in \mathbb{R}^{n\times d}$ with each row vector constructed by method as shown
   in Eq. (3.2) or Eq. (3.3)
2: Draw $\mathbf{\Theta} \in \mathbb{R}^{n\times \ell}$ from $\mathcal{N}(0, 1)$
3: Construct $\mathbf{Y} = (\mathbf{F}^T\mathbf{F})^q\mathbf{F}^T\mathbf{\Theta}$
4: Run QR decomposition on $\mathbf{Y}$ such that $\mathbf{QR} = \mathbf{Y}$ where $\mathbf{Q} \in \mathbb{R}^{d\times \ell}$ is
   column-orthogonal
5: return $\mathbf{G} = \mathbf{FQ} \in \mathbb{R}^{n\times \ell}$
```

We summarize this method in Algorithm 3.1. As can be seen, there are two parts in this algorithm. Step 1 represents a feature matrix $\mathbf{F}$, where the inverse Fourier transform $p(\mathbf{z})$ of different $k(\cdot)$ can be found in [153]. Steps 2 to 5 show the FDSE. q is typically set by 0, 1 or 2, which is used to speed up the decay of the singular value of $\mathbf{F}$ and thus improves the accuracy of the whole algorithm [79].

Moreover, step 4 in Algorithm 3.1 can still be accelerated by replacing $\mathbf{\Theta}$ with SRHT (Subsampled Randomized Hadamard Transform), then running QR decomposition only requires $O(nd \log \ell + d\ell^2)$ time. For the details of SRHT, please refer to [170]. We summarize this method in Algorithm 3.2 that runs faster than Algorithm 3.1.

```
Algorithm 3.2 TEFM-S
Input: Data $\mathbf{X}^T = \{\mathbf{x}_i \in \mathbb{R}^m\}_{i=1}^n$, shift-invariant kernel function $k(\cdot)$,
    scalars $[d,\, \ell]$
1: Run step 1 of Algorithm 3.1
2: Draw a SRHT matrix $\mathbf{\Theta} \in \mathbb{R}^{n\times \ell}$
3: Construct $\mathbf{Y} = \mathbf{F}^T\mathbf{\Theta}$
4: Run steps 4, 5 of Algorithm 3.1
5: return $\mathbf{G} \in \mathbb{R}^{n\times \ell}$
```

3.3.2 Kernel Approximation Analysis

In this part, we provide theoretical analysis to show that $O(\ell)$ features generated by our algorithms can approximate kernel matrix as accurately as $O(\ell^2)$ features induced by other existing methods like RKS and its extensions. Below, we present our main results.

Theorem 3.1. *Suppose we have a kernel matrix* $\mathbf{K} \in \mathbb{R}^{n\times n}$ *constructed from shift-invariant functions and run Algorithm 3.1 with* $d = \Omega(\ell^2)$ *to get features* $\mathbf{G} \in \mathbb{R}^{n\times\ell}$*. Then, with probability at least* $1 - \delta - 6\exp^{-p}$ *we have*

$$\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2 \leq \widetilde{O}(n/\ell), \tag{3.4}$$

where $\widetilde{O}(\cdot)$ *hides the logarithmic factors on* δ *and the power coefficient on* d *and* p *that tend to be* 1*.*

Apparently, this result Eq. (3.4) is tighter than its counterpart $\widetilde{O}(n/\sqrt{\ell})$ [127, 153]. As shown from its proof in Section 3.6, Eq. (3.4) results from Eq. (3.12), i.e., $\widetilde{O}(n/\sqrt{d}) + O(n/\ell)$, which is incurred by RKS and FDSE. The first term of Eq. (3.12) denotes a spectral norm bound for RKS by Eq. (3.2) or Eq. (3.3), which is slightly tighter and more general than the expectation bound in [127], and reflects the same convergence rate on RKS feature number as the element-wise bound [153]. If $d = \Omega(\ell^2)$, then training on $O(\ell)$ feature generated by our two algorithms will be much faster than $O(\ell^2)$ features by RKS with the approximation error almost on the same scale. In the experiments, we will see the accuracy versus the computational time for the training and feature map.

Remark 3.1. The work in [81] tries to approximate the polynomial kernel more concisely and accurately, by which our approach is motivated similarly but has important differences. First, we use FDSE instead of random projection to approximate more accurately while maintaining the computational efficiency. Second, the theoretical result therein shows that the extent of approximation improvement relies on the degree of the polynomial kernel, while ours depends on the number of finally generated features, which directly demonstrates that we can use *much fewer informative* features to train. Third, the experiment shows that our method can achieve improvement in the shift-invariant kernels, while [81] even *degenerates* the performance, which we conjecture is due to the different sampling procedures for the feature maps.

Theorem 3.2. *Suppose we have a kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ constructed from shift-invariant functions and run Algorithm 3.2 with $d = \Omega(\ell^2)$ to obtain features $\mathbf{G} \in \mathbb{R}^{n\times \ell}$. Then, with probability at least $1 - \delta - \frac{c}{\ell \log \ell}$ we have*

$$\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2 \leq \widetilde{O}(n/\ell), \tag{3.5}$$

where c is a certain constant, and $\widetilde{O}(\cdot)$ hides the logarithmic factors on δ, d, and n.

Usually, the failure probability can be larger than that in Theorem 3.1, which can be checked in the proof in Section 3.6. This comparison implies Algorithm 3.1 may perform more accurately than Algorithm 3.2 although it runs slower.

3.3.3 Impact on Learning Tasks

In this part, we show how our features impact the learning accuracy on regression and classification tasks. The related algorithms including KRR and SVM can be unified into the following optimization problem [193, 194],

$$\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} p\{\hbar(\mathbf{w}^T\mathbf{Z}(\mathbf{x}_i), y_i)\}, \tag{3.6}$$

where $p(t)$ means convex loss functions such as $p(t) = t^2/2$, $p(t) = \log\{1+\exp(-t)\}$ and $p(t) = \max(0, 1-t)$. That $t = \hbar(\mathbf{w}^T\mathbf{Z}(\mathbf{x}_i), y_i)$ denotes $\mathbf{w}^T\mathbf{Z}(\mathbf{x}_i) - y_i$ or $\mathbf{w}^T\mathbf{Z}(\mathbf{x}_i)y_i$. We also assume the magnitude of gradient $p'(t)$ can be upper bounded by C (nondifferentiable loss function like $p(t) = \max(0, 1-t)$ in SVM can be made differentiable by smoothing techniques [143, 201]).

Theorem 3.3. *Suppose we get kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ by operating shift-invariant functions on data matrix $\mathbf{X}^T = \{\mathbf{x}_i \in \mathbb{R}^m\}_{i=1}^n$, and feature matrix $\mathbf{G}^T = \{\mathbf{G}_i \in \mathbb{R}^\ell\}_{i=1}^n$ by Algorithm 3.1 or Algorithm 3.2 with $d = \Omega(\ell^2)$. Denote by $L(\mathbf{w}^*)$ the optimal value of Eq. (3.6). Let $L(\mathbf{w}^*_{\mathbf{G}})$ be the obtained optimal value by training on $\mathbf{Z}(\mathbf{x}_i) = \mathbf{G}_i$, and $L(\mathbf{w}^*_{\mathbf{K}})$ the optimal value by by training on $\mathbf{K}$ (i.e., $\mathbf{Z}(\mathbf{x}_i) = \Phi(\mathbf{x}_i)$), where $\mathbf{w}^*_{\mathbf{G}} \in \mathbb{R}^\ell$ and $\mathbf{w}^*_{\mathbf{K}}$ may be infinite-dimensional. Then, with probability defined in Theorem 3.1 or Theorem 3.2 we have*

$$L(\mathbf{w}^*_{\mathbf{G}}) \le L(\mathbf{w}^*_{\mathbf{K}}) + \widetilde{O}(1/\ell), \tag{3.7}$$

where $\widetilde{O}(\cdot)$ hides the logarithmic factors on δ, d, and n.

We can extend the above result to other features with different kernel approximation performances, showing that a faster convergence of the kernel approximation error can lead to faster convergence in the minimized regularized training error. This can be used as a simplified measurement to quantify the impact of kernel approximation on the learning accuracy [32, 41]. Therefore, applying the features that ensure a good kernel approximation favors an accurate learning on the learning tasks.

3.3.4 Computational Analysis

In this part, we compare the computational performance on ridge regression task written in the form of Eq. (3.6) using the square loss function. We use ridge regression due to that it can give a closed-form solutions for the mapped features and standard kernel, then the computational time does not depend on some specific optimization techniques.

We summarize the result in Table 5.1, where nnz$(\cdot)$ means the number of non-zero value and t the number of test points. As we can see, with $d = O(\ell^2)$ set, our two algorithms take much less time, and also significantly reduce the the overall time with the dominant training time decreased.

For RKS, only a 'seed' is needed for random number generator to reproduce random sequence for the testing [44]. After getting $\mathbf{w}$ by Eq. (3.6), we practically multiply it with $\mathbf{Q}$ of Algorithm 3.1 or Algorithm 3.2 for the testing. Finally, our algorithms only stores a 'seed' and a vector $\mathbf{Q}\mathbf{w}^T$ for the testing, taking comparable storage with other methods.

3.4 Empirical Studies

In this section, we empirically demonstrate the superiority of the proposed methods on the Gaussian kernel matrix approximation and related regression task. We compare the following random feature map methods:

1. Random Kitchen Sinks (denoted by RKS).

Table 3.1: Time complexity of $\mathcal{M}_1$ to $\mathcal{M}_4$ representing 4 methods respectively, i.e., Kernel method, RKS, TEFM-G and TEFM-S.

	Mapping	**Training**	**Prediction**
$\mathcal{M}_1$	$O(\mathrm{nnz}(\mathbf{X})n)$	$O(n^3)$	$O(t(m)n)$
$\mathcal{M}_2$	$O(\mathrm{nnz}(\mathbf{X})d)$	$O(nd^2)$	$O(t(m+1)d)$
$\mathcal{M}_3$	$O(\mathrm{nnz}(\mathbf{X})d$ $+\ell^2 d + n\ell d)$	$O(n\ell^2)$	$O(t(m+1)d$ $+d\ell)$
$\mathcal{M}_4$	$O(\mathrm{nnz}(\mathbf{X})d$ $+\ell^2 d + nd\log\ell)$	$O(n\ell^2)$	$O(t(m+1)d$ $+d\ell)$

2. Quasi-Monte Carlo method [197] (denoted by Quasi). We use Digital Net to generate QMC sequence, as it yields the lowest approximation error and supports the high dimensional data [197].

3. Compact feature maps [81] (denoted by Comp). The proposed method should not be restricted to the polynomial kernel, we thus apply it to Gaussian kernel in our experiments.

4. Fastfood method [109] (denoted by Ffood). We use *Hadamard features* for its better performance.

5. Our proposed algorithms TEFM-G and TEFM-S.

We use six publicly available real-world datasets listed in Table 3.2, and they can be downloaded from LIBSVM website [33] or UCI machine learning repository. All our experiments are run on Matlab with single thread mode in order to fairly compare the running time.

3.4.1 Kernel Approximation Quality

We compare the features generated by above methods on the kernel approximation. We report the approximation accuracy of each method measured by $\|\mathbf{K} - \mathbf{K}_{\text{app}}\|/\|\mathbf{K}\|$, where $\mathbf{K}_{\text{app}}$ is

Table 3.2: Details of datasets in our experiments.

Dataset	**# instances**	**# features**
Cpu	6554	21
A9a	48,842	123
BlogFeedback	60,021	280
SliceLocalization	53,500	384
UJIIndoorLoc	21,048	520
Mnist	70,000	784

the approximation matrix reconstructed by the random features. To facilitate the computation of full kernel matrix $\mathbf{K}$ for the comparison purposes, we randomly sample these datasets so that only 6554, 8141, 8733, 8917, 6646, and 6006 instances are used, respectively.

We summarize the results in Figure 3.1. The x-axis stands for the number of features ℓ. In our two algorithms, typically approximation error will turn down as d or ℓ increases, however, d cannot be very large or even infinite, otherwise our two feature generation algorithms will be time consuming. Thus, d actually affects the tradeoff between the accuracy and computation. If we keep $d = 4\ell$, then Figure 3.1 shows that our algorithms incur nearly *half* of approximation error compared to others, which almost coincides with our theoretical result. This implies that it is possible to train the learning tasks with fewer features while maintaining the accuracy. In particular, our methods beat the counterpart *Comp* which implements data-independent subspace embedding for original features [81]. This also reflects that data-independent subspace embedding is not suitable for the random features on shift-invariant kernels.

We also observe the oscillatory nature in Figure 3.1. The reason relies on that the randomness incurs the variance (os-

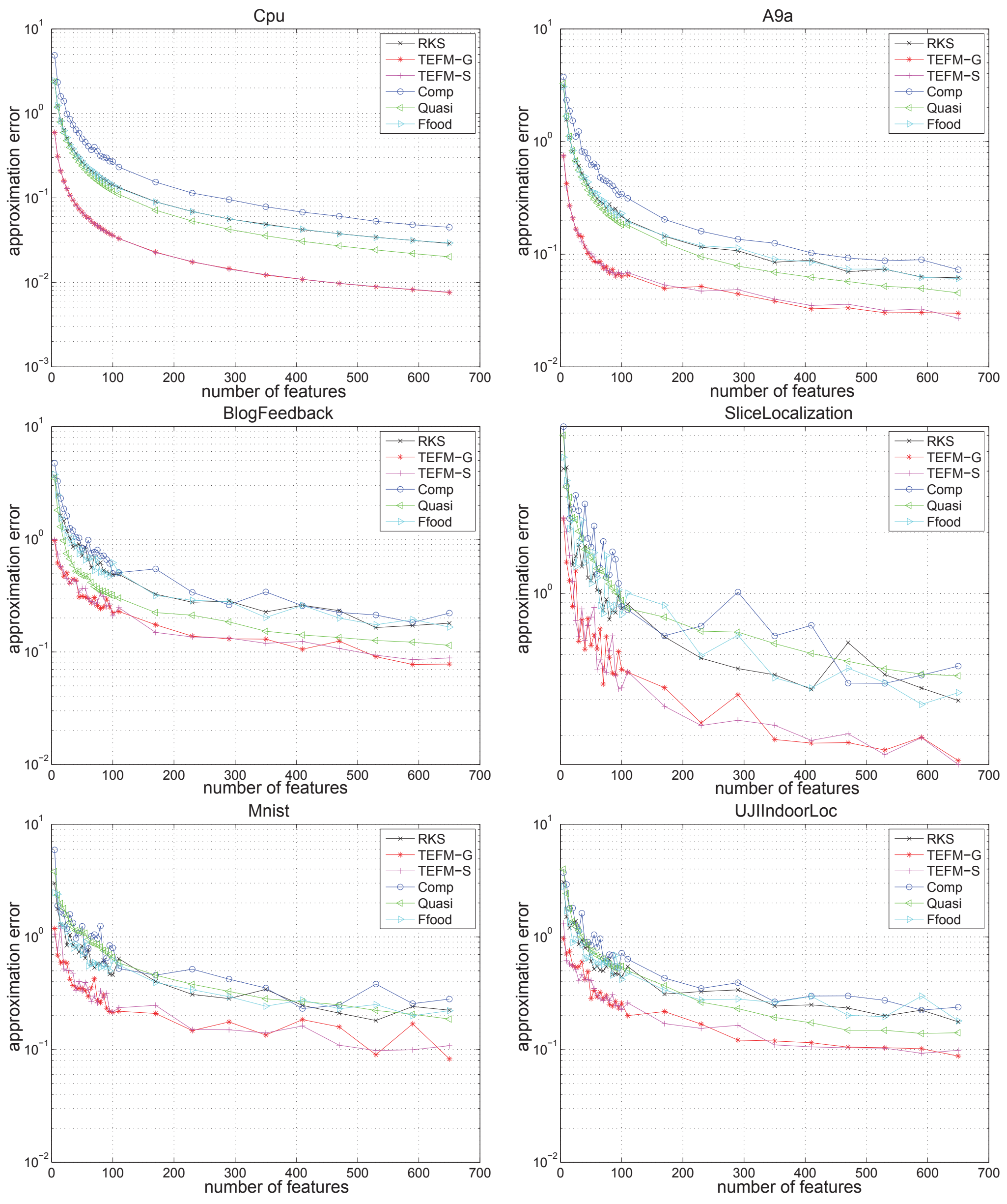

Figure 3.1: Kernel approximation.

cillatory), and the kernel matrix approximation is based on the average of the randomness. When the number of random feature is small, we cannot make a stable kernel matrix approximation by averaging the approximated kernel matrices from each random feature. Thus, it is easy to observe the oscillatory nature, and it can also be observed in the empirical results of Quasi-Monte Carlo feature map [197].

3.4.2 Performance on KRR

First, we compare the prediction ability of different features on KRR. The parameters are chosen by cross-validation. We report the relative root mean square error (RMSE) of each method in the testing datasets. The relative RMSE is defined by $\|\mathbf{y}-\mathbf{y}_o\|/\|\mathbf{y}\|$, where $\mathbf{y}_o$ is the predicted value and $\mathbf{y}$ is the ground truth. We consider the regression dataset SliceLocalization and list the result in the left plot of Figure 3.2. As we can see, using much fewer informative features, our two algorithms can obtain the same accuracy as others. Thus, training using our fewer features can keep the learning accuracy and require much less time. This also implies that, accurate learning can be achieved by applying the features that ensure a good kernel matrix approximation.

Second, we report the prediction error versus the time (*includes feature mapping time and training time*) in the right plot of Figure 3.2. Our algorithms require slightly more feature map time. They, however, use fewer features to get the same approximation performance. Therefore, this substantially decreases the training time. As can be seen, even we consider the time used for the feature map, our two algorithms still outperform the other methods with smaller RMSE achieved.

Particularly, the advantage of our methods will be more evident when we need a much smaller RMSE. The reason is that, to obtain a much smaller RMSE, more features in other methods should be generated, and as the feature number grows large, the time used for training regression task will be dominant compared to the time for feature map.

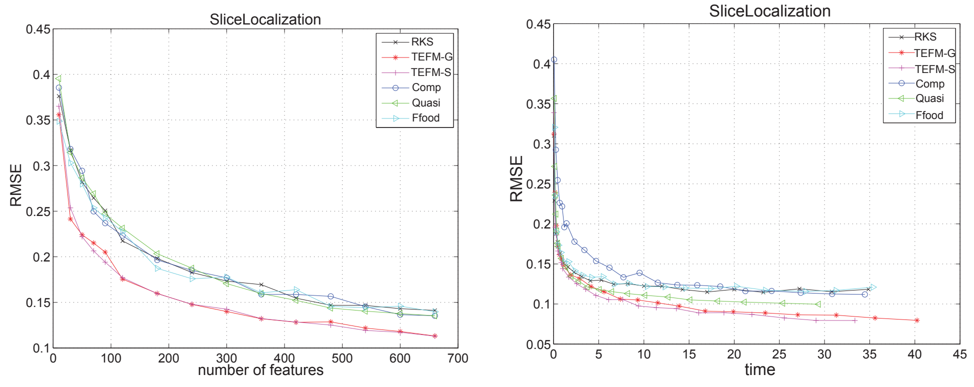

Figure 3.2: Accuracy comparision and time comparison on regression.

3.5 Conclusion

In this chapter, we describe an effective feature map method that can generate better features to make the learning tasks trained accurately and more efficiently. The basic idea of our method is to combine fast data-dependent subspace embedding with the RKS. We also present two algorithms TEFM-G and TEFM-S. Our theoretical analysis indicates these two algorithms achieve better kernel approximation and faster training on learning tasks without losing precision. We report extensive empirical results of our algorithms on several real-world datasets, which support our analysis and demonstrate a good practical performance.

3.6 Proofs

3.6.1 Preliminaries

Before proceeding, we first state two extracted and reformulated lemmas. They are our main tools to derive the theoretical results.

Lemma 3.4 ([171]). *Suppose* $\{\mathbf{S}_i\}_{i=1}^n$ *are random square matrices with* $\mathbb{E}[\mathbf{S}_i] = \mathbf{0}$ *and* $\|\mathbf{S}_i\|_2 \leq L$. *Define* $\mathbf{Z} = \sum_{i=1}^n \mathbf{S}_i$, Ξ_1 *and* Ξ_2 *satisfying* $\mathbb{E}[\mathbf{Z}\mathbf{Z}^T] \preccurlyeq \Xi_1$ *and* $\mathbb{E}[\mathbf{Z}^T\mathbf{Z}] \preccurlyeq \Xi_2$. *Let* $\xi = \max\{\|\Xi_1\|_2, \|\Xi_2\|_2\}$ *and* $r = \mathrm{Tr}(\Xi_1 + \Xi_2)/\xi$. *Then for* $t \geqslant \sqrt{\xi} + L/3$, *we have*

$$\mathbb{P}\{\|\mathbf{Z}\|_2 \geqslant t\} \leq 4r\exp(\frac{-t^2/2}{\xi + Lt/3}). \tag{3.8}$$

Lemma 3.5 ([79]). *Denote the SVD of* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *by* $\mathbf{A} = \mathbf{U}\mathbb{D}(\mathbf{\Sigma}_1, \mathbf{\Sigma}_2)(\mathbf{V}_1, \mathbf{V}_2)^T$ *where* $\mathbf{\Sigma}_1 \in \mathbb{R}^{k\times k}$, $\mathbf{\Sigma}_2 \in \mathbb{R}^{(n-k)\times(n-k)}$, $\mathbf{V}_1 \in \mathbb{R}^{n\times k}$, $\mathbf{V}_2 \in \mathbb{R}^{n\times(n-k)}$ *and* $\mathbb{D}(\mathbf{\Sigma}_1, \mathbf{\Sigma}_2) \in \mathbb{R}^{n\times n}$ *with* $\mathbf{\Sigma}_1$ *and* $\mathbf{\Sigma}_2$ *on its main diagonal. Given* $\mathbf{\Omega} \in \mathbb{R}^{n\times l}$ *and construct* $\mathbf{Y} = (\mathbf{A}\mathbf{A}^T)^q\mathbf{A}\mathbf{\Omega}$. *If* $\mathbf{V}_1^T\mathbf{\Omega}$ *has full row rank, then we have*

$$\begin{aligned}\|(\mathbf{I}-\mathbf{Y}\mathbf{Y}^\dagger)\mathbf{A}\|_2^2 &\leq \|(\mathbf{I}-\mathbf{Y}\mathbf{Y}^\dagger)(\mathbf{A}^T\mathbf{A})^q\mathbf{A}\|_2^{2/(2q+1)} \\ &\leq (\|\mathbf{\Sigma}_2\|_2^{4q+2} + \|\mathbf{\Sigma}_2^{4q+2}(\mathbf{V}_2^T\mathbf{\Omega})(\mathbf{V}_1^T\mathbf{\Omega})^\dagger\|_2^2)^{1/(2q+1)}. \end{aligned} \tag{3.9}$$

3.6.2 Proof of Theorem 3.4

Proof. Without loss of generality, we analyze the case in detail where $\mathbf{F}$ in step 1 corresponds to Eq. (3.2) and Gaussian kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/2\sigma^2)$. The theorem and analysis can be applied in other cases. Revisit the variables and parameters in Algorithm 3.1, we have

$$\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2 = \|\mathbf{K} - \mathbf{F}\mathbf{Q}\mathbf{Q}^T\mathbf{F}^T\|_2$$

$$\begin{aligned}
&\le \|\mathbf{K} - \mathbf{F}\mathbf{F}^T\|_2 + \|\mathbf{F}\mathbf{F}^T - \mathbf{F}\mathbf{Q}\mathbf{Q}^T\mathbf{F}^T\|_2 \\
&= \|\mathbf{K} - \mathbf{F}\mathbf{F}^T\|_2 + \|\mathbf{F}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)^2\mathbf{F}^T\|_2 \qquad (3.10)\\
&= \|\mathbf{K} - \mathbf{F}\mathbf{F}^T\|_2 + \|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\mathbf{F}^T\|_2^2 \qquad (3.11)\\
&\le O(n/\sqrt{d}) + O(n/\ell), \qquad (3.12)
\end{aligned}$$

where Eq. (3.10) follows from the fact $(\mathbf{I}-\mathbf{Q}\mathbf{Q}^T)$ is an orthogonal projection matrix and Eq. (3.12) will be proved as follows.

The first term in Eq. (3.12) is proved by Lemma 3.4. Define $a = d/2$ and $\mathbf{H}_i \in \mathbb{R}^{n\times 2}$ formed by columns $2i-1$ and $2i$ of $\mathbf{F}$, i.e., $\{\cos(\mathbf{z}_i^T\mathbf{x}_j)/\sqrt{a}, \sin(\mathbf{z}_i^T\mathbf{x}_j)/\sqrt{a}\}_{j=1}^n \in \mathbb{R}^{n\times 2}$, $\forall i \in [a]$.

Let $\mathbf{S}_i = (\mathbf{K} - a\mathbf{H}_i\mathbf{H}_i{}^T)/a$. Then, $\mathbf{K} - \mathbf{F}\mathbf{F}^T = \sum_{i=1}^a \mathbf{S}_i$ and $\mathbb{E}[\mathbf{S}_i] = \mathbf{0}$. Also, $\mathbb{E}_{i\neq j}[\mathbf{S}_i^T\mathbf{S}_j] = \mathbf{0}$ holds by that $\{\mathbf{H}_i\}_{i=1}^a$ are independent. Moreover, $\|\mathbf{K}\|_2 \le n$ due to $\mathbf{K}_{jj} \le 1$, and $\|\mathbf{H}_i\|_2^2 = \|\mathbf{H}_i\mathbf{H}_i^T\|_2 \le (\sqrt{1/a})^2 n = n/a$. Thus, $\|\mathbf{S}_i\|_2 \le 2n/a = L$.

Following the notations in Lemma 3.4, we accordingly define $\mathbf{\Xi}_1 = \mathbf{\Xi}_2 = \mathbb{E}[\mathbf{Z}^T\mathbf{Z}] = \mathbb{E}[(\sum_{i=1}^a \mathbf{S}_i)^T(\sum_{i=1}^a \mathbf{S}_i)] = a\mathbb{E}[\mathbf{S}_i^T\mathbf{S}_i]$ since $\mathbf{Z} = \sum_{i=1}^a \mathbf{S}_i$ is symmetric and $\mathbb{E}_{i\neq j}[\mathbf{S}_i^T\mathbf{S}_j] = \mathbf{0}$. Then, we have

$$\begin{aligned}
\xi = \|\mathbf{\Xi}_1\|_2 &= \|a\mathbb{E}[(\mathbf{K} - a\mathbf{H}_i\mathbf{H}_i^T)^2/a^2]\|_2 \\
&= \|\{a^2\mathbb{E}[(\mathbf{H}_i\mathbf{H}_i^T)^2] - \mathbf{K}^2\}/a\|_2 \\
&= \|\{a^2\mathbb{E}[\mathbf{H}_i\mathbf{H}_i^T\mathbf{H}_i\mathbf{H}_i^T] - \mathbf{K}^2\}/a\|_2 \\
&\le \|\{a^2\mathbb{E}[\|\mathbf{H}_i\|_2^2\mathbf{H}_i\mathbf{H}_i^T] - \mathbf{K}^2\}/a\|_2 \qquad (3.13)\\
&\le \|(n\mathbf{K} - \mathbf{K}^2)/a\| \\
&= \|\{\mathbf{U}(n\mathbf{\Sigma} - \mathbf{\Sigma}^2)\mathbf{U}^T\}/a\|_2 \\
&= \|(n\mathbf{\Sigma} - \mathbf{\Sigma}^2)/a\|_2 \\
&\le n^2/(4a), \qquad (3.14)
\end{aligned}$$

where $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ is the SVD of $\mathbf{K}$ with the eigenvalues $\Sigma_{ii} = \sigma_i$ listed in the descending order.

In the above inequalities, Eq. (3.13) holds by

$$\begin{aligned} 0 \preccurlyeq \boldsymbol{\Xi}_1 &= \{a^2\mathbb{E}[\mathbf{H}_i\mathbf{H}_i^T\mathbf{H}_i\mathbf{H}_i^T] - \mathbf{K}^2\}/a \\ &\preccurlyeq \{a^2\mathbb{E}[\|\mathbf{H}_i\|_2^2\mathbf{H}_i\mathbf{H}_i^T] - \mathbf{K}^2\}/a, \end{aligned} \tag{3.15}$$

due to that for any $\mathbf{x} \in \mathbb{R}^n$

$$\begin{aligned} \mathbf{x}^T\mathbf{H}_i\mathbf{H}_i^T\mathbf{H}_i\mathbf{H}_i^T\mathbf{x} &= \|\mathbf{x}^T\mathbf{H}_i\mathbf{H}_i^T\|_2^2 \\ &\leq \|\mathbf{x}^T\mathbf{H}_i\|^2\|\mathbf{H}_i^T\|_2^2 \\ &= \mathbf{x}^T\mathbf{H}_i\mathbf{H}_i^T\mathbf{x}\|\mathbf{H}_i^T\|_2^2. \end{aligned} \tag{3.16}$$

Eq. (3.14) follows from that

$$\max_{0\leq\sigma_i\leq n} |(n\sigma_i - \sigma_i^2)/a| = n^2/(4a). \tag{3.17}$$

In addition to that $r = \text{Tr}(\boldsymbol{\Xi}_1 + \boldsymbol{\Xi}_2)/\xi$ usually can satisfy that $1 \leq r \ll O(n)$, we can get the concentration bound

$$\mathbb{P}(\|\mathbf{K} - \mathbf{F}\mathbf{F}^T\|_2 \geqslant t) \leq 4r\exp\{\frac{-t^2/2}{n^2/(4a) + 2nt/(3a)}\}. \tag{3.18}$$

Let $\delta = 4r\exp\{\frac{-t^2/2}{n^2/(4a)+2nt/(3a)}\}$ and $\theta = 4r/\delta$. Then, we have

$$\begin{aligned} \|\mathbf{K} - \mathbf{F}\mathbf{F}^T\|_2 &\leq \ln\theta\{2n/3a + \sqrt{4n^2/9a^2 + n^2/(2a\ln\theta)}\} \\ &= \widetilde{O}(n/\sqrt{a}) = \widetilde{O}(n/\sqrt{d}), \end{aligned} \tag{3.19}$$

which holds with failure probability at most δ, and we can regard $\ln\theta$ as a small value and also easily check that $t \geqslant \sqrt{\xi} + L/3$ holds.

To prove the second term in Eq. (3.12), we use Lemma 3.5 and get

$$\begin{aligned} \|(\mathbf{I}-\mathbf{Q}\mathbf{Q}^T)\mathbf{F}^T\|_2^2 &\leq \|(\mathbf{I} - \mathbf{Y}\mathbf{Y}^\dagger)\mathbf{F}^T\|_2^2 \qquad (3.20) \\ &\leq (\|\boldsymbol{\Sigma}_2\|_2^{4q+2} + \|\boldsymbol{\Sigma}_2^{4q+2}(\mathbf{V}_2^T\boldsymbol{\Theta})(\mathbf{V}_1^T\boldsymbol{\Theta})^\dagger\|_2^2)^{1/(2q+1)} \end{aligned}$$

$$\leq (1 + \|(\mathbf{V}_2^T \boldsymbol{\Theta})(\mathbf{V}_1^T \boldsymbol{\Theta})^\dagger\|_2^2)^{1/(2q+1)} \|\boldsymbol{\Sigma}_2\|_2^2, \tag{3.21}$$

where $\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{U}\mathbb{D}(\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)(\mathbf{V}_1, \mathbf{V}_2)^T$ with $\boldsymbol{\Sigma}_1 \in \mathbb{R}^{k\times k}$ and $\boldsymbol{\Sigma}_2 \in \mathbb{R}^{(d-k)\times(d-k)}$, and Eq. (3.20) holds because $\text{span}(\mathbf{Y}) \subset \text{span}(\mathbf{Q})$.

Notice that $\boldsymbol{\Theta}$ is a centered Gaussian matrix, thus $\widehat{\mathbf{V}}_1^T \boldsymbol{\Theta}$ has full row rank with probability one and Lemma 3.5 can be applied. The derivation for the upper bound of $\|(\mathbf{V}_2 \boldsymbol{\Theta})(\mathbf{V}_1^T \boldsymbol{\Theta})^\dagger\|_2$ follows from [79]. Then, combining and reformulating them, we get that $\|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\mathbf{F}^T\|_2^2 \leq [k, q, d, p]^{1/(2q+1)} n/(k+1) = O(n/\ell)$ holds with failure probability at most $6\exp(-p)$, where $\ell = k+p$ with p typically set to be a small constant (e.g., 5), $[k, q, d, p]$ means an expression on variables k, q, d, p, which turns smaller as p increases and is briefly represented here due to limited space. $[k, q, d, p]^{1/(2q+1)}$ can be considered as a small value if with a slightly big q.

By union bound and setting $d = \Omega(\ell^2)$, we prove Eq. (3.4) holds with failure probability at most $\delta + 6\exp(-p)$. □

3.6.3 Proof of Theorem 3.5

Proof. The proof is similar to that in Theorem 3.1 by setting $q = 0$ in Eq. (3.21). Then, with failure probability at most c/k, $\|(\mathbf{V}_2 \boldsymbol{\Theta})(\mathbf{V}_1^T \boldsymbol{\Theta})^\dagger\|_2$ has a constant upper bound if $\ell = Ck \log k$, where c and C are some constants [170]. Treating $\log k$ as a small value and using $\|\boldsymbol{\Sigma}_2\|_2^2 \leq n/(k+1)$ can complete the proof.

Note that q has to be zero and c/k can be larger than $6\exp(-p)$ in Theorem 3.1. This implies Algorithm 3.1 may perform more accurately than Algorithm 3.2 although it runs slower. □

3.6.4 Proof of Theorem 3.7

Proof. Instead of directly applying Representer Theorem [159] on Eq. (3.6), a desired tight result can be obtained using its Lagrangian duality. Rewrite Eq. (3.6) into a constrained convex optimization problem by introducing new variables $\mathbf{g}^T = \{g_i = \hbar(\mathbf{w}^T\mathbf{Z}(\mathbf{x}_i), y_i)\}_{i=1}^n \in \mathbb{R}^{1\times n}$, and combine the Lagrangian with KKT conditions, then we get an equivalent dual problem without duality gap, i.e.,

$$L(\mathbf{w}^*) = \max_{\boldsymbol{\alpha}} \sum_{i=1}^n p_*(\alpha_i) - \frac{1}{2\lambda}\boldsymbol{\alpha}^T\mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^T\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{y}, \quad (3.22)$$

where $\boldsymbol{\alpha}^T = \{\alpha_i\}_{i=1}^n \in \mathbb{R}^{1\times n}$, $\mathbf{y}^T = \{y_i\}_{i=1}^n \in \mathbb{R}^{1\times n}$, each row of $\mathbf{Z}(\mathbf{X})$ is formed by $\mathbf{Z}(\mathbf{x}_i)$, $\hbar(\mathbf{w}^T\mathbf{Z}(\mathbf{x}_i), y_i) = \mathbf{w}^T\mathbf{Z}(\mathbf{x}_i) - y_i$ without loss of generality, and $p_*(\alpha_i) = \inf\{-\boldsymbol{\alpha}^T\mathbf{g} + \sum_{i=1}^n p(g_i)/n\}$ is supposed to be finite with $\alpha_i = p'(g_i)/n$, $\forall i \in [n]$. Then,

$$\begin{aligned} L(\mathbf{w}^*_{\mathbf{G}}) &= \max_{\boldsymbol{\alpha}} \sum_{i=1}^n p_*(\alpha_i) - \frac{1}{2\lambda}\boldsymbol{\alpha}^T\mathbf{G}\mathbf{G}^T\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{y} \\ &\le L(\mathbf{w}^*_{\mathbf{K}}) + \max_{\boldsymbol{\alpha}} \frac{1}{2\lambda}\boldsymbol{\alpha}^T(\mathbf{K} - \mathbf{G}\mathbf{G}^T)\boldsymbol{\alpha} \\ &\le L(\mathbf{w}^*_{\mathbf{K}}) + \frac{1}{2\lambda}\|\boldsymbol{\alpha}\|\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2\|\boldsymbol{\alpha}\|_2 \\ &\le L(\mathbf{w}^*_{\mathbf{K}}) + \frac{n(C/n)^2}{2\lambda}\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2 \\ &\le L(\mathbf{w}^*_{\mathbf{K}}) + \frac{C^2}{2n\lambda}O(\frac{n}{\ell}) = L(\mathbf{w}^*_{\mathbf{K}}) + \widetilde{O}(\frac{1}{\ell}), \end{aligned}$$

where λ is assumed to be a certain fixed value and the last inequality holds with limited failure probability by using Theorem 3.1 or Theorem 3.2. □

□ **End of chapter.**

Chapter 4

Faster Online Sketching Hashing

Many hashing methods, especially those that are in the data-dependent category with good learning accuracy, are still inefficient when dealing with three critical problems in modern data analysis. First, data usually come in a streaming fashion, but most of the existing hashing methods are batch-based models. Second, when data become huge, the extensive computational time, large space requirement, and multiple passes to load the data into memory will be prohibitive. Third, data often lack sufficient label information. Although the recently proposed Online Sketching Hashing (OSH) is promising to alleviate all three issues mentioned above, its training procedure still suffers from a high time complexity. In this chapter, we propose a FasteR Online Sketching Hashing (FROSH) method to make the training process faster. Compared with OSH, we leverage fast JL transform to sketch data more compactly. Particularly, we derive independent transformations to guarantee the sketching accuracy, and design a novel implementation to make such transformations applicable to online data sketching without increasing the space cost. We rigorously prove that our method can yield a comparable learning accuracy with a lower time complexity and an equal space cost compared with OSH. Finally, extensive experiments on synthetic and real-world datasets demonstrate the excellent performance of our method.

4.1 Introduction

Hashing is an efficient method to conduct an approximate nearest neighbor search, which is critical for machine learning and applications such as clustering, retrieval and matching [177]. It transforms data into a low-dimensional representation, i.e., a short hash code consisting of a sequence of bits in Hamming space. The data distance in original space then is approximated by the Hamming distance that can be calculated extremely fast in modern CPU. Thus, the approximate nearest neighbor search can be accomplished with less time and space costs. Current hashing methods can be categorized broadly as data-independent and data-dependent techniques. Locality Sensitive Hashing (LSH) methods [13, 34, 72, 94, 165] are prime examples of data-independent techniques, which are also unsupervised. They construct hash functions based on random projection, which are typically very fast and theoretically guaranteed, but are developed only for certain distance functions and often require long code length to achieve acceptable search accuracy because of ignoring the data distribution. Compared with data-independent methods, data-dependent hashing techniques achieve better accuracy performance with shorter binary codes, while usually incurring a larger computational cost to train the hashing functions. These data-dependent methods can be categorized as unsupervised and (semi-)supervised techniques.

In unsupervised studies [77, 97, 111, 123, 124, 125, 139, 162, 185, 199], hash functions are learned from data distribution rather than being randomly generated by preserving a metric induced distance in the Hamming space. Supervised methods [102, 116, 117, 121, 144, 163, 175] additionally leverage the label information, and thus often outperform those unsupervised ones.

Although data-dependent hashing methods have achieved a promising learning accuracy, yet they suffer from some critical problems when confronted with the data such as web images, videos, stocks, genomes and web documents in the big data era. First, data often become available continuously in a streaming fashion, and each data point or data chunk can be processed only once [111, 118]. Hence, batch learning strategies are not allowed. Moreover, with batch-learners, it is unclear how to adapt the hash functions as the dataset continues to grow and new variations appear over time. Second, the data size n and dimension d can be large with $1 \ll d \ll n$ [53], so that the high time complexity and $O(nd)$ space cost [111, 118] are prohibitive. In particular, advanced batch-based unsupervised hashing solutions such as SGH [97] and OCH [123] can avoid the $O(nd)$ space cost merely by performing multiple passes over the data, while the disk IO overhead for loading all data into memory multiple times will be the major performance bottleneck [190]. Third, labels are commonly missing, noisy, and scarce in today's big data situation, and labeling streaming data are also expensive and infeasible [166, 189, 191].

To tackle the above challenges, we focus on the most recently proposed online hashing, i.e., online sketching hashing (OSH) [111], which is data-dependent, space-efficient, unsupervised, and in a single pass. Note that there have been several other investigations in online hashing. Such hashing methods include online kernel-based hashing (OKH) [92], adaptive online hashing (AOH) [29], and [30] but they all belong to the supervised category, which require extra label information.

However, given $O(d\ell)$ space to store and perform calculation on the streaming data, the OSH method retains a training time of $O(nd\ell + d\ell^2)$ to yield $r \leq O(\ell)$ hashing bits, where n is the data size, d is the data dimension, and ℓ denotes the sketching size satisfying $\ell < d \ll n$. Such training time is still expensive since $1 \ll d \ll n$ [53]. In this chapter, we attempt to reduce the running time further. Our contributions are summarized as follows:

- First, we propose a FasteR Online Sketching Hashing (FROSH) method by improving the data sketching procedure in the OSH method. Specifically, we employ a fast JL transform to reduce the data size, in which independent transformations are applied for different small data chunks to make the sketching compact and accurate.

- Second, we design a more efficient way to implement the fast JL transform in our FROSH. Compared with the standard way, our strategy reduces the space burden incurred by the fast JL transform for sketching streaming data while maintaining the same time efficiency.

- Third, we analyze the accuracy of our FROSH, and give an error bound comparable with OSH. Moreover, our FROSH has a smaller time complexity of $\widetilde{O}(n\ell^2 + nd)$ with an equal space usage. Extensive experiments demonstrate the computational efficiency with a competitive learning accuracy.

The remainder of this chapter is organized as follows. In Section 2, we review the prior work and techniques. In Section 3, we present our method along with theoretical analysis, and emphasize its advantages from various aspects. In Section 4, we provide extensive empirical results, and Section 5 concludes the whole work.

4.2 Related Work

In this section, we describe the properties of the OSH method. Then, we outline and discuss the fast JL transforms, especially the Subsampled Randomized Hadamard Transform (SRHT).

4.2.1 Online Sketching Hashing

We rephrase the background and details of the OSH method [111]. Given data $\mathbf{A} \in \mathbb{R}^{n\times d}$ and the unknown projection matrix $\mathbf{W} \in \mathbb{R}^{d\times r}$, the k-th hashing function for a data point $\mathbf{a}^i \in \mathbb{R}^{1\times d}$ is defined as

$$h_k(\mathbf{a}^i) = \mathrm{sgn}((\mathbf{a}^i - \boldsymbol{\mu})\mathbf{w}_k), \tag{4.1}$$

where $\boldsymbol{\mu} = \bar{\mathbf{A}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{a}^i$. By dropping the non-differentiable function $\mathrm{sgn}(\cdot)$ for the binary codes [175], the objective can be reformulated as the same as that of Principal Component Analysis (PCA)

$$\begin{aligned} \max_{\mathbf{W}\in\mathbb{R}^{d\times r}} \quad & \mathrm{Tr}(\mathbf{W}^T(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu})\mathbf{W}) \\ \text{s.t.} \quad & \mathbf{W}^T\mathbf{W} = \mathbf{I}_r, \end{aligned} \tag{4.2}$$

where $(\mathbf{A}-\boldsymbol{\mu})$ denotes a matrix $[\mathbf{a}^1-\boldsymbol{\mu}; \mathbf{a}^2-\boldsymbol{\mu}; \ldots; \mathbf{a}^n-\boldsymbol{\mu}]$. The

Algorithm 4.1 Online Sketching Hashing (OSH) [111]

```
Input: Data $\mathbf{A} = \{\mathbf{A}_j \in \mathbb{R}^{h_j \times d}\}_{j=1}^{s}$, sketching size $\ell < d$, positive integer $\eta$, hashing bits $r$
1: Initialize sketching matrix by $\mathbf{B} = \mathbf{0}^{\ell \times d}$
2: Set $\boldsymbol{\mu}_1$ as the row mean vector of $\mathbf{A}_1$
3: Let $\boldsymbol{\varphi} = \boldsymbol{\mu}_1$, $\tau = h_1$, $\xi = 0$
4: Sketch $\mathbf{G}_1 = (\mathbf{A}_1 - \boldsymbol{\mu}_1) \in \mathbb{R}^{h_1 \times d}$ into $\mathbf{B} \in \mathbb{R}^{\ell \times d}$
5: for $j = 2 : s$ do
6:     Set $\boldsymbol{\mu}_j$ as the row mean vector of $\mathbf{A}_j$
7:     Set $\boldsymbol{\varsigma} = \sqrt{\frac{\tau h_j}{\tau + h_j}}(\boldsymbol{\mu}_j - \boldsymbol{\varphi}) \in \mathbb{R}^{1 \times d}$
8:     Sketch $\mathbf{G}_j = [(\mathbf{A}_j - \boldsymbol{\mu}_j); \boldsymbol{\varsigma}] \in \mathbb{R}^{(h_j+1) \times d}$ into $\mathbf{B} \in \mathbb{R}^{\ell \times d}$
9:     Set $\boldsymbol{\varphi} = \frac{\tau \boldsymbol{\varphi}}{\tau + h_j} + \frac{h_j \boldsymbol{\mu}_j}{\tau + h_j}$ # Update the mean vector
10:    Set $\tau = \tau + h_j$ # Update the data size
11:    Set $\xi = \xi + 1$
12:    if $\xi == \eta$ then
13:        Compute the SVD of $\mathbf{B} \in \mathbb{R}^{\ell \times d}$, and assign the top $r$ right singular vectors as $\mathbf{W}^T \in \mathbb{R}^{r \times d}$
14:        Set $\xi = 0$
15:        return W
16:    end if
17: end for
```

solution to $\mathbf{W} \in \mathbb{R}^{d \times r}$ in Eq. (4.2) is the top r right eigenvectors of the covariance matrix $(\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu})$. For $d < n$, it takes $O(nd^2)$ time and $O(nd)$ space, which is infeasible for large n and d [101].

To tackle above computational issues, sketching the data before the training is a promising way. Specifically, a sketch of a data matrix is another matrix that is significantly smaller than the original but still approximates it well and preserves the properties of interest. It implies that the storing and the computing on the sketch will be much easier than with the original large matrix, and the downstream learning algorithms on the sketch can still guarantee the learning accuracy [15, 40, 118, 129, 188].

Leveraging the effectiveness of sketching in the learning, OSH is proposed, and its details are presented in Algorithm 4.1. It aims at efficiently achieving a small matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ for the centered data $(\mathbf{A} - \boldsymbol{\mu})$, then computing the SVD on $\mathbf{B}$ only takes $O(d\ell^2)$ time and $O(d\ell)$ space for $\ell < d$. The difficulty lies in that sketching cannot be directly applied in each centered data chunk $(\mathbf{A}_j - \boldsymbol{\mu})$, since $\boldsymbol{\mu}$ can only be obtained after observing all data points in $\mathbf{A}$.

Then, the *online centering procedure* in steps 3 to 10 are to ensure that $(\mathbf{A}_{[j]} - \hat{\boldsymbol{\mu}}_j)^T(\mathbf{A}_{[j]} - \hat{\boldsymbol{\mu}}_j) = \mathbf{G}_{[j]}^T\mathbf{G}_{[j]}$ after the j-th iteration, where $\mathbf{A}_{[j]} = [\mathbf{A}_1; \mathbf{A}_2; \cdots; \mathbf{A}_j]$ by stacking all observed $\mathbf{A}_j$ vertically, $\mathbf{G}_{[j]}$ follows a similar definition, and $\hat{\boldsymbol{\mu}}_j$ is the row mean vector of $\mathbf{A}_{[j]}$. Steps 4 and 8 are to get a smaller matrix $\mathbf{B}$ via an efficient sketching such that $\mathbf{B}^T\mathbf{B} \approx \mathbf{G}_{[j]}^T\mathbf{G}_{[j]}$ after the j-th iteration. After all iterations, OSH guarantees that $\mathbf{B}^T\mathbf{B} \approx \mathbf{G}_{[s]}^T\mathbf{G}_{[s]} = (\mathbf{A}_{[s]} - \hat{\boldsymbol{\mu}}_s)^T(\mathbf{A}_{[s]} - \hat{\boldsymbol{\mu}}_s) = (\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu})$.

If $\sum_{j=1}^{s} h_j = O(n)$, the sketching steps in OSH will take $O(nd\ell)$ time and $O(d\ell)$ space. The SVD in step 13 runs for about ξ/η times. As ξ/η can be manually set to be a small constant, step 13 takes $O(d\ell^2)$ time and $O(d\ell)$ space in total. The computational cost of the online data centering procedure is negligible. In conclusion, OSH consumes $O(nd\ell + d\ell^2)$ time and $O(d\ell)$ space.

Remark 4.1. To address the problem that most of the information can be contained by only a small number of significant singular vectors in $\mathbf{W} \in \mathbb{R}^{d \times r}$, OSH also empirically applies a random orthogonal rotation $\boldsymbol{\Upsilon} \in \mathbb{R}^{r \times r}$ (the orthonormal bases of an $r \times r$ random Gaussian matrix) to all singular vectors $\mathbf{W} \in \mathbb{R}^{d \times r}$ in Algorithm 4.1 via $\mathbf{W}\boldsymbol{\Upsilon}$. This step resembles Iterative Quantization [77] but runs much more efficiently with streaming settings maintained and negligible computational cost incurred.

4.2.2 Frequent Directions for Sketching

Algorithm 4.2 Frequent Directions (FD) [118]

```
Input: Data $\mathbf{A} \in \mathbb{R}^{n \times d}$, sketching matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$
1: if $\mathbf{B}$ not exists then
2:     Set $\mathbf{B} = \mathbf{0}^{\ell \times d}$
3: end if
4: for $i \in [n]$ do
5:     Insert $\mathbf{a}^i \in \mathbb{R}^{1 \times d}$ into a zero valued row of $\mathbf{B}$
6:     if $\mathbf{B}$ has no zero valued rows then
7:         $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \mathrm{SVD}(\mathbf{B})$
8:         $\widehat{\mathbf{\Sigma}} = \sqrt{\max(\mathbf{\Sigma}^2 - \sigma_{\ell/2}^2 \mathbf{I}_l, 0)}$
9:         Set $\mathbf{B} = \widehat{\mathbf{\Sigma}} \mathbf{V}^T$
10:    end if
11: end for
```

OSH employs the Frequent Directions (FD) as a black-box to sketch streaming data, which is summarized in Algorithm 4.2. It operates by keeping collecting row vectors from the data source. Once the sketching matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ has ℓ non-zero rows, the shrinking procedure as shown in steps 6 to 9 will reduce the size of the matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ by a half, *which repeats throughout the entire streaming data.*

In this part, we assume that FD will process n data points. The computational cost of the FD algorithm is dominated by the shrinking procedure that involves computing an exact SVD on $\mathbf{B} \in \mathbb{R}^{\ell \times d}$, which takes $O(d\ell^2)$ time and $O(d\ell)$ space for $\ell < d$. Since the shrinking procedure is operated once nearly every $\ell/2$ iterations of the main loop, the time complexity is $O(d\ell^2 \times n/\ell) = O(nd\ell)$ with $O(d\ell)$ space burden.

Recently, the Sparse FD (SFD) algorithm [70] is proposed to take advantage of the sparsity of $\mathbf{A}$ by applying a powerful randomized SVD [140] instead of an exact SVD in each shrinking procedure. SFD still takes $O(d\ell)$ space but only requires $\widetilde{O}(\text{nnz}(\mathbf{A})\ell + n\ell^2)$ running time, where $\text{nnz}(\mathbf{A})$ means the number of non-zero entries in $\mathbf{A}$. This time cost is much smaller than $O(nd\ell)$ if $\ell \ll d$ and $\mathbf{A} \in \mathbb{R}^{n\times d}$ is extremely sparse.

Remark 4.2. FD works well for streaming data. For instance, given two (or more) data chunks like $\mathbf{G}_1 \in \mathbb{R}^{h_1\times d}$ and $\mathbf{G}_2 \in \mathbb{R}^{(h_2+1)\times d}$ in Algorithm 4.1, we invoke FD and run it on $\mathbf{G}_1$ to yield a sketching matrix $\mathbf{B} = \mathbf{B}_1 \in \mathbb{R}^{\ell\times d}$. Based on the current $\mathbf{B} = \mathbf{B}_1$, we then run FD on $\mathbf{G}_2$ to update $\mathbf{B}$ and get $\mathbf{B} = \mathbf{B}_2 \in \mathbb{R}^{\ell\times d}$. Such procedures are obviously equivalent to that we directly invoke FD for once and run it on $[\mathbf{G}_1; \mathbf{G}_2] \in \mathbb{R}^{(h_1+h_2+1)\times d}$, which also gets an identical $\mathbf{B} = \mathbf{B}_2$.

4.2.3 Fast JL Transform

For the convenience of reference, we restate the fast JL transform as described previously in Chapter 2. Given a vector $\mathbf{a} \in \mathbb{R}^m$, its compressed representation $\mathbf{b} \in \mathbb{R}^q$ can be obtained by performing the matrix-vector multiplication like $\mathbf{\Phi a}$, where $q < m$ and $\mathbf{\Phi} \in \mathbb{R}^{q\times m}$ is a random projection matrix (e.g., a Gaussian random matrix). Generally, computing $\mathbf{\Phi a}$ takes $O(qm)$ time. It can be a computational bottleneck of fast implementing numerous learning tasks that involve data compression.

Fast JL transforms, which are based on the structured projection matrix like Hadamard matrix or Fourier matrix, then are employed to overcome the shortcomings of the classical transformation methods. For Hadamard matrix $\mathbf{H}_m \in \mathbb{R}^{m\times m}$, its entry is defined as

$$\mathbf{H}_{ij} = (-1)^{\langle i-1, j-1 \rangle}, \tag{4.3}$$

where $\langle i-1, j-1 \rangle$ is the dot-product of the b-bit vectors of integers $i-1$ and $j-1$ expressed in binary, and $b \leq \max(\lceil \log(i+1) \rceil, \lceil \log(j+1) \rceil)$. It can also be defined recursively as

$$\mathbf{H}_m = \begin{bmatrix} \mathbf{H}_{m/2} & \mathbf{H}_{m/2} \\ \mathbf{H}_{m/2} & -\mathbf{H}_{m/2} \end{bmatrix} \quad \text{and} \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The normalized Hadamard matrix is defined by $\mathbf{H} = \sqrt{\frac{1}{m}}\mathbf{H}_m$, and by exploring the recursive structure of $\mathbf{H}$, computing $\mathbf{Ha}$ only takes $O(m \log m)$ time with $O(m)$ space [7].

To compress data, we perform the Subsampled Randomized Hadamard Transform (SRHT), i.e., data are transformed via a randomized Hadamard matrix and then uniformly sampled in the resulting entries. For a $q \times m$ SRHT matrix, it is defined as the form $\mathbf{\Phi} = \mathbf{SHD}$, where $\mathbf{D} \in \mathbb{R}^{m \times m}$ is a diagonal matrix with its diagonal elements being i.i.d. Rademacher random variables (i.e., 1 or -1 with equal probability), $\mathbf{S} \in \mathbb{R}^{q \times m}$ is a scaled sampling matrix with each row uniformly sampled *without replacement* from m rows of the identity matrix $\mathbf{I}_m \in \mathbb{R}^{m \times m}$ multiplied by $\sqrt{\frac{m}{q}}$.

Differently, computing $\mathbf{\Phi a}$ only takes $O(m \log q)$ time with $O(m)$ space [7, 128].

4.3 Faster Online Sketching Hashing

In this section, we motivate and present FROSH, whose sketching relies on our Faster FD (FFD). To make FROSH maintain an equal space efficiency with that of OSH, we also derive a new implementation of fast JL transform in FFD for sketching streaming data. The analysis guarantees the performance of our solution.

4.3.1 Motivation, Method and Results

The computational cost of OSH is dominated by its sketching method FD. This sketching requires $O(nd\ell)$ time to obtain $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ from the data $\mathbf{A} \in \mathbb{R}^{n \times d}$, which is computationally expensive for $1 \ll d \ll n$ [53]. Even regarding SFD, its $\widetilde{O}(\text{nnz}(\mathbf{A})\ell + nl^2)$ time cost still tends to be $\widetilde{O}(nd\ell + n\ell^2)$ since the centered data $\mathbf{A} - \boldsymbol{\mu}$ are dense.

We first present an FFD sketching method as stated in Algorithm 4.3 to tackle above issues. In steps 6 to 9, after collecting m data points into $\mathbf{F} \in \mathbb{R}^{m \times d}$, a new SRHT matrix is generated to compress $\mathbf{F}$ with its size reduced to $\ell/2$. Then, the newly compressed data and previously shrunken data form the $\mathbf{B} \in \mathbb{R}^{\ell \times d}$. Steps 10 to 11 employ a shrinking procedure that is similar to that in FD. However, there may exist one problem in each iteration of step 9, i.e., computing $\mathbf{\Phi}_t\mathbf{F}$ in the standard way has to take $O(md \log \ell)$ time but with $O(md)$ space to collect all m data points. For a big $m = \Theta(d)$, the space cost of $O(md) = O(d^2)$ is practically prohibitive since space can be severely limited [118], which is also inferior to FD method whose advantages includes that only $O(d\ell)$ space is employed.

Fortunately, we do not have to keep a matrix $\mathbf{F} \in \mathbb{R}^{m \times d}$ explicitly to store m data points. Instead, $\mathbf{F} \in \mathbb{R}^{m \times d}$ here is simply used for easily presenting the algorithm. Furthermore, as data come sequentially, we only take $O(d\ell)$ space to handle $O(\ell)$ data points, save and combine intermediate results. We

Algorithm 4.3 Faster Frequent Directions (FFD)

```
Input: Data $\mathbf{A} \in \mathbb{R}^{n\times d}$, sketching matrix $\mathbf{B} \in \mathbb{R}^{\ell\times d}$
 1: if $\mathbf{B}$ not exists then
 2:     Set $t = 1$, $\mathbf{B} = \mathbf{0}^{\ell\times d}$
 3:     Set $\mathbf{F} = \mathbf{0}^{m\times d}$ with $m = \Theta(d)$ # Only needed for notation
 4: end if
 5: for $i \in [n]$ do
 6:     Insert $\mathbf{a}^i$ into a zero valued row of $\mathbf{F}$
 7:     if $\mathbf{F}$ has no zero valued rows then
 8:         Generate a fast JL transform $\mathbf{\Phi}_t = \mathbf{S}_t\mathbf{H}\mathbf{D}_t \in \mathbb{R}^{(\ell/2)\times m}$ in the $t$-th
            trial
 9:         Insert $\mathbf{\Phi}_t\mathbf{F} \in \mathbb{R}^{(\ell/2)\times d}$ into the last $\frac{\ell}{2}$ rows of $\mathbf{B}$
10:         $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \mathrm{SVD}(\mathbf{B})$
11:         $\widehat{\mathbf{\Sigma}} = \sqrt{\max(\mathbf{\Sigma}^2 - \sigma^2_{\ell/2}\mathbf{I}_l, 0)}$
12:         Set $\mathbf{B} = \widehat{\mathbf{\Sigma}}\mathbf{V}^T$
13:         Let $\mathbf{B}_t = \mathbf{B}$, $\mathbf{C}_t = \mathbf{\Phi}_t\mathbf{F}$ # Only needed for proof notations
14:         Set $\mathbf{F} = \mathbf{0}^{m\times d}$, $t = t + 1$
15:     end if
16: end for
```

repeat such procedures until that we have sequentially processed m data points denoted by $\mathbf{F} \in \mathbb{R}^{m\times d}$, which finally still only takes $O(md\log\ell)$ time in total to achieve $\mathbf{\Phi}_t\mathbf{F}$. The detailed implementation is deferred to the subsequent section.

Moreover, we adopt an independent and distinct $\mathbf{\Phi}_t$ in each iteration of step 9, which benefits the sketching accuracy by controlling the variance in all sketchings. Below, we formally characterize the properties of our proposed FFD.

Theorem 4.1 (FFD). *Given data $\mathbf{A} \in \mathbb{R}^{n\times d}$ and the sketching size $\ell \leq k = \min(m, d)$, let the small sketch $\mathbf{B} \in \mathbb{R}^{\ell\times d}$ be constructed by FFD. Then, with probability at least $1 - p\beta -$*

$(2p+1)\delta - \frac{2n}{e^k}$ we have

$$\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2 \le \widetilde{O}\Big(\frac{1}{\ell} + \Gamma(\ell,p,k)\Big)\|\mathbf{A}\|_F^2 \tag{4.4}$$

where $\Gamma(\ell,p,k) = \sqrt{\frac{k}{\ell p^2}} + \sqrt{\frac{1+\sqrt{k/\ell}}{p}}$ with $p = \frac{n}{m}$, and $\widetilde{O}(\cdot)$ hides logarithmic factors on (β,δ,k,d,m) as in Eqs. (4.21)(4.25)(4.40).

The running time of the algorithm is $\widetilde{O}(nl^2\frac{d}{m} + nd)$ and its space cost is $O(d\ell)$ before taking $m = \Theta(d)$.

Table 4.1: Time cost and error bounds of the compared frequent direction methods, where the space cost is $O(d\ell)$ for all methods.

Method	**Time**	**Error Bound**
FD [118]	$O(nd\ell)$	$\frac{2}{\ell}\|\mathbf{A}\|_F^2$
SFD [70]	$O(\mathrm{nnz}(\mathbf{A})\ell + nl^2)$ $\big(O(nd\ell + n\ell^2)$ for dense centers$\big)$	$\widetilde{O}(\frac{1}{\ell}\|\mathbf{A}\|_F^2)$
FFD	$\widetilde{O}(\frac{nl^2d}{m} + nd)$ $\big(\widetilde{O}(n\ell^2 + nd)$ when $m = \Theta(d)\big)$	$\widetilde{O}(\frac{1}{\ell} + \Gamma(\ell,p,k))\|\mathbf{A}\|_F^2$ $\big(\widetilde{O}(\frac{1}{\ell}\|\mathbf{A}\|_F^2)$ when $p = \Omega(\ell^{3/2}k^{1/2})\big)$

Remark 4.3. For a fixed n, a smaller m improves the sketching accuracy while taking more time burden. We choose $m = \Theta(d)$ such that the time consumption of FFD is $\widetilde{O}(n\ell^2 + nd)$. If $\ell \ll d$, it is superior to $O(nd\ell)$ time cost in FD and $\widetilde{O}(nd\ell + n\ell^2)$ time usage in SFD for dense centered data.

When data keep coming (i.e., n increases) and/or m decreases, p will get larger, which makes the bound in Eq. (4.4) tighter. If $p = \Omega(\ell^{3/2}k^{1/2})$, i.e., $n = \Omega(\ell^{3/2}d^{3/2})$ when $m = \Theta(d)$, then the error bound of FFD becomes $\widetilde{O}(\frac{1}{\ell}\|\mathbf{A}\|_F^2)$, which is asymptotically comparable with the bounds $\frac{2}{\ell}\|\mathbf{A}\|_F^2$ of FD and $\widetilde{O}(\frac{1}{\ell}\|\mathbf{A}\|_F^2)$ of SFD.

Thus, FFD is more applicable to the big data situation $1 \ll d \ll n$ with computational cost reduced and accuracy guaranteed. Competitive empirical results suggest there is an opportunity to further tighten our error bound. The detailed comparisons can be found in Table 4.1

Algorithm 4.4 FasteR Online Sketching Hashing (FROSH)

```
Input: Data $\mathbf{A} = \{\mathbf{A}_j \in \mathbb{R}^{h_j \times d}\}_{j=1}^{s}$, sketching size $\ell < d$, positive integer $\eta$, hashing bits $r$
1: Run steps 1 to 3 of Algorithm 4.1
2: Run steps 4 to 10 of Algorithm 4.1 with FFD to sketch the centered data into $\mathbf{B} \in \mathbb{R}^{\ell \times d}$
3: Run steps 11 to 15 of Algorithm 4.1 and return $\mathbf{W} \in \mathbb{R}^{d \times r}$
```

Based on FFD and OSH, our FROSH is straightforward given in Algorithm 4.4. Its step 2 contains the online data centering procedure, which is equivalent to that in OSH. Then, combining Theorem 4.1 and the properties of the online data centering procedure yields the next result.

Theorem 4.2 (FROSH)**.** *Given data $\mathbf{A} \in \mathbb{R}^{n \times d}$ with its row mean vector $\boldsymbol{\mu} \in \mathbb{R}^{1 \times d}$, let the sketch $\mathbf{B}^{\ell \times d}$ be generated by FROSH in Algorithm 4.4. Then, with probability defined in Theorem 4.1 we have*

$$\begin{aligned} &\|(\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu}) - \mathbf{B}^T\mathbf{B}\|_2 \\ &\qquad \leq \widetilde{O}\Big(\frac{1}{\ell} + \Gamma(\ell, p, k)\Big)\|\mathbf{A} - \boldsymbol{\mu}\|_F^2, \end{aligned} \tag{4.5}$$

where $(\mathbf{A} - \boldsymbol{\mu}) \in \mathbb{R}^{n \times d}$ means subtracting each row of $\mathbf{A}$ by $\boldsymbol{\mu}$, $\Gamma(\ell, p, k)$ is from Theorem 4.1, and the top r right singular vectors of $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ are used for hashing projections $\mathbf{W}^T \in \mathbb{R}^{r \times d}$ in Remark 4.1.

The algorithm takes $\widetilde{O}(n\ell^2 + nd + d\ell^2)$ time and $O(d\ell)$ space cost after taking $m = \Theta(d)$ in the FFD of FROSH.

Remark 4.4. This primary analysis resembles that of OSH [111], which follows the conception that accurate sketching does not harm the learning accuracy [15, 40, 111, 118, 129, 188]. Hence, the $\mathbf{W}$ from $\mathbf{B}$ is expected to well approximate that from $\mathbf{A} - \boldsymbol{\mu}$ (more illustrations on the approximation for $\mathbf{W}$ are deferred into the appendix), and we use this sketching error for the centered data $\mathbf{A} - \boldsymbol{\mu}$ to justify if the sketching-based hashing works properly [111]. Thus, the accuracy comparisons between OSH and FROSH resemble Remark 4.3. Compared with $\widetilde{O}(nd\ell+d\ell^2)$ time and $O(d\ell)$ space cost in OSH, we improve a lot if $1 < \ell \ll d \ll n$.

Next, we offer more details to show how the project matrix $\mathbf{W}^T \in \mathbb{R}^{r\times d}$ can be approximated. We let $m = \Theta(d)$, and assume $n = \Omega(\ell^{3/2}d^{3/2})$ for simplicity, then the error bound of Eq. (5) in Theorem 4.2 becomes $\widetilde{O}(\frac{1}{\ell}\|(\mathbf{A} - \boldsymbol{\mu})\|_F^2)$. Based on it, we give Corollary 4.1.

Corollary 4.1. *Given data $\mathbf{A} \in \mathbb{R}^{n\times d}$ with its row mean vector $\boldsymbol{\mu} \in \mathbb{R}^{1\times d}$, let the sketching matrix $\mathbf{B}^{\ell\times d}$ be generated by FROSH in Algorithm 4.4. Let $m = \Theta(d)$, and assume $n = \Omega(\ell^{3/2}d^{3/2})$ for simplicity. Given $(\mathbf{A} - \boldsymbol{\mu}) \in \mathbb{R}^{n\times d}$ that means subtracting each row of $\mathbf{A}$ by $\boldsymbol{\mu}$, let $h = \|(\mathbf{A} - \boldsymbol{\mu})\|_F^2/\|(\mathbf{A} - \boldsymbol{\mu})\|_2^2$ and σ_i be the i-th largest singular value of $(\mathbf{A} - \boldsymbol{\mu})$. If the sketching size $\ell = \Omega(\frac{h\sigma_1^2}{\epsilon\sigma_{r+1}^2})$, then with probability defined in Theorem 1 we have*

$$\begin{aligned}&\|(\mathbf{A} - \boldsymbol{\mu}) - (\mathbf{A} - \boldsymbol{\mu})\mathbf{W}_{\mathbf{B}}\mathbf{W}_{\mathbf{B}}^T\|_2^2 \\ &\qquad \leq (1+\epsilon)\|(\mathbf{A} - \boldsymbol{\mu}) - (\mathbf{A} - \boldsymbol{\mu})\mathbf{W}\mathbf{W}^T\|_2^2, \qquad (4.6)\end{aligned}$$

where $0 < \epsilon < 1$, $\mathbf{W}_{\mathbf{B}}^T \in \mathbb{R}^{r\times d}$ contains the top r right singular vectors of $\mathbf{B}^{\ell\times d}$, and $\mathbf{W}^T \in \mathbb{R}^{r\times d}$ contains the top r right singular vectors of $(\mathbf{A} - \boldsymbol{\mu}) \in \mathbb{R}^{n\times d}$.

Remark 4.5. The bound on $\|(\mathbf{A}-\boldsymbol{\mu})-(\mathbf{A}-\boldsymbol{\mu})\mathbf{W_B}\mathbf{W_B}^T\|_2^2$ shows the similarity between $\mathbf{W_B}\mathbf{W_B}^T$ and $\mathbf{W}\mathbf{W}^T$. If $\epsilon = 0$, we will have $\mathbf{W_B}\mathbf{W_B}^T = \mathbf{W}\mathbf{W}^T$. However, it cannot characterize the similarity between $\mathbf{W_B} \in \mathbb{R}^{d\times r}$ and $\mathbf{W} \in \mathbb{R}^{d\times r}$, because Eq. (4.6) of Corollary 4.1 may also indicate that $\mathbf{W_B}$ approximates $\mathbf{W}\boldsymbol{\Upsilon}$, where $\boldsymbol{\Upsilon} \in \mathbb{R}^{r\times r}$ is an arbitrary unitary matrix with $\boldsymbol{\Upsilon}\boldsymbol{\Upsilon}^T = \mathbf{I}_r$ and $\mathbf{I}_r$ being an identity matrix so that $\mathbf{W}\boldsymbol{\Upsilon}\boldsymbol{\Upsilon}^T\mathbf{W}^T = \mathbf{W}\mathbf{W}^T$. Fortunately, due to that $\boldsymbol{\Upsilon}\boldsymbol{\Upsilon}^T = \mathbf{I}_r$ (i.e., $\boldsymbol{\Upsilon} \in \mathbb{R}^{r\times r}$ is an orthogonal rotation), $\mathbf{W}\boldsymbol{\Upsilon}$ will still retain all information of $\mathbf{W}$ and even empirically get better hashing accuracy, which has been mentioned in Remark 4.1. Therefore, Corollary 4.1 shows how $\mathbf{W_B}$ approximates $\mathbf{W}$ or $\mathbf{W}\boldsymbol{\Upsilon}$, which can be used to show the effectiveness of the related hashing algorithm.

4.3.2 Implementation of the Fast JL Trasform in FFD

Define $q = \ell/2$, then we show how to simply adopt $O(qd)$ space and perform a single pass through $\mathbf{F} \in \mathbb{R}^{m\times d}$ to achieve $\mathbf{\Phi F}$ with $O(md\log q)$ time, given $\mathbf{\Phi} = \mathbf{SHD} \in \mathbb{R}^{q\times m}$, $q \leq m$, and $\log m \leq O(d\log q)$.

To clarify, we choose an m such that $m = 2^b$ for a positive integer b. We can always find a $q/2 \leq p \leq q$ to define positive integers c and p such that $m = 2^c p$.

As shown in Figure 4.1, we divide data matrix $\mathbf{F} \in \mathbb{R}^{m\times d}$ into 2^c blocks as $\{\mathbf{F}_i \in \mathbb{R}^{p\times d}\}_{i=1}^{2^c}$, and the diagonal matrix $\mathbf{D} \in \mathbb{R}^{m\times m}$ into 2^c square blocks as $\{\mathbf{D}_i \in \mathbb{R}^{p\times d}\}_{i=1}^{2^c}$. Hadamard matrix $\mathbf{H} \in \mathbb{R}^{m\times m}$ can also be divided into $(2^c)^2 = 4^c$ square blocks $\{\mathbf{H}_{ij} \in \mathbb{R}^{p\times p}\}_{i,j=1}^{2^c}$.

We take $O(pd)$ space to receive streaming data from $\mathbf{F}_1 \in \mathbb{R}^{p\times d}$. Then, through $\mathbf{S}[\mathbf{H}_{11};\mathbf{H}_{21};\cdots;\mathbf{H}_{2^c1}]\mathbf{D}_1\mathbf{F}_1 \in \mathbb{R}^{q\times d}$ we perform the matrix multiplication. We first run $\mathbf{H}_{11}\mathbf{D}_1\mathbf{F}_1 \in$

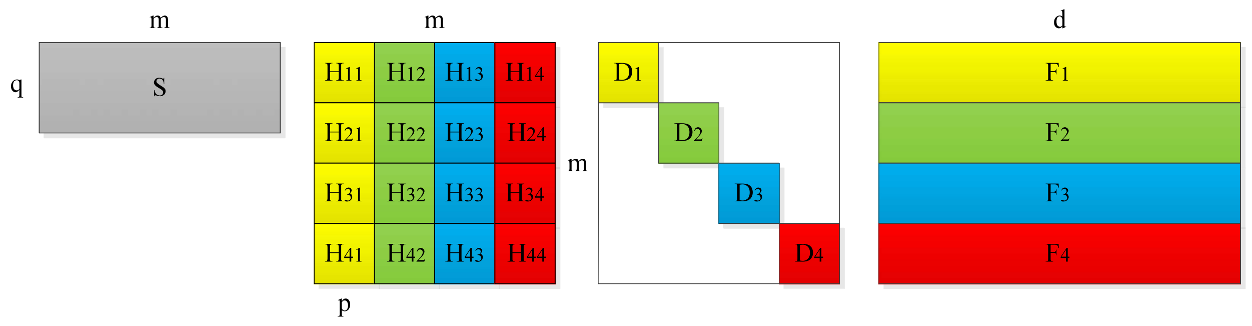

Figure 4.1: Space-efficient implementation of $\mathbf{\Phi F} = \mathbf{SHDF}$. We set $m/p = 2^c$ with $c = 2$ for clarity.

$\mathbb{R}^{p\times d}$, taking $O(pd \log p)$ time and $O(pd)$ space. We then only have to get at most q resulting rows from $\{\mathbf{H}_{i1}\mathbf{F}_1\}_{i\geq 2}^{2^c}$ since $\mathbf{S} \in \mathbb{R}^{q\times m}$ only selects q rows. Fortunately, $\mathbf{H}_{11} = +\mathbf{H}_{i1}$ or $\mathbf{H}_{11} = -\mathbf{H}_{i1}$ holds for any $i \in [2^c]$, and the $\pm$ sign can be determined by calculating the first entry of each $\mathbf{H}_{i1}$ based on Eq. (4.3) whose running time is at most $O(\log m)$. Hence, the ultimate time complexity is $O(pd \log p) + O(qd) + O(q \log m)$. It is also straightforward to check that $O(pd + qd)$ space suffices for all calculations and saving the compressed data.

Then, we remove $\mathbf{F}_1$, receive data from $\mathbf{F}_2$, do the similar calculations, and update current result based on the previous compressed data from $\mathbf{F}_1$. The time and space costs on $\mathbf{F}_2$ still keep asymptotically unchanged.

Finally, the space usage of computing $\mathbf{\Phi F}$ is $O(pd) = O(qd)$, and its time cost is $2^c[O(pd \log p) + O(qd) + O(q \log m)] = O(md \log q)$ given $q/2 \leq p \leq q$, $m = 2^c p$, and $\log m \leq O(d \log q)$.

4.4 Empirical Studies

In this section, we conduct three sets of experiments to empirically verify the properties and demonstrate the superiority of our proposed FROSH. The experiments are conducted in MATLAB R2015a and run on a standard workstation with Intel CPU@2.90GHz and 128GB RAM. The MATLAB is set in single thread mode to fairly test time. All results are averaged over 10 independent runs.

4.4.1 Numerical Studies on Sketching

First of all, it is necessary to verify the sketching properties of FD and FFD, which dominate the performance of OSH and FROSH, respectively. We aim to demonstrate that FFD can achieve a comparable sketching accuracy with a faster running speed in comparison with FD.

To make an insightful comparison, we run both algorithms on the synthetic data $\mathbf{A} \in \mathbb{R}^{n\times d}$, which is generated as [118] to verify the sketching methods. Specifically, $\mathbf{A} \in \mathbb{R}^{n\times d}$ is formally defined as $\mathbf{A} = \mathbf{GFU} + \mathbf{W}/\gamma$, where $\mathbf{G} \in \mathbb{R}^{n\times k}$ is the signal coefficient with $g_{ij} \sim \mathcal{N}(0,1)$, the square diagonal matrix $\mathbf{F} \in \mathbb{R}^{k\times k}$ contains the diagonal entries $f_{ii} = 1 - (i-1)/k$ that gives linearly diminishing signal singular values, $\mathbf{U} \in \mathbb{R}^{k\times d}$ defines the signal row space with $\mathbf{U}\mathbf{U}^T = \mathbf{I}_k$ ($k \ll d$), and $\mathbf{W} \in \mathbb{R}^{n\times d}$ consists of noise $w_{ij} \sim \mathcal{N}(0,1)$.The parameter γ determines if the noise can dominate the signal. Following [118], we set $k = 10$ and $\gamma = 10$.

In the experiments, we vary the sketching size ℓ along $(16, 32, 64, 100, 128, 200, 256)$. We also vary the data size n and dimension d of $\mathbf{A} \in \mathbb{R}^{n\times d}$, and the parameter m in FFD. Such variables can impact the sketching performance in different degrees.

We plot the relative error $\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2/\|\mathbf{A}\|_F^2$ versus the sketching size ℓ in Figure 4.2 and compare the running

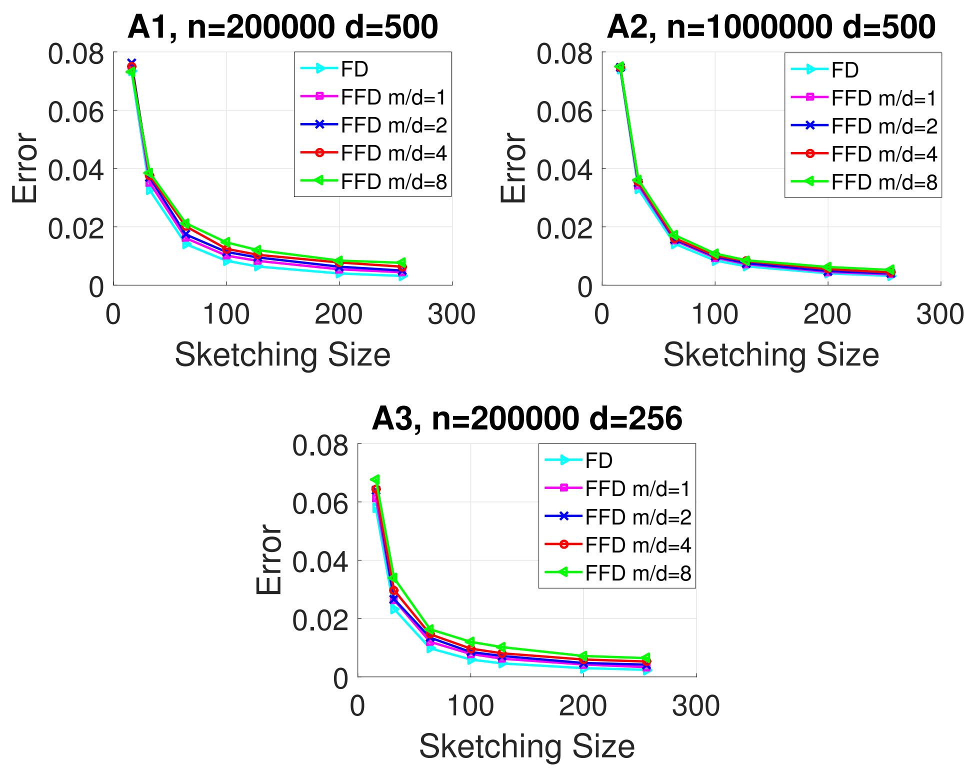

Figure 4.2: Accuracy comparisons of sketching methods.

time given in Figure 4.3. It can be seen that, for data $\mathbf{A}_1$, FFD displays comparable accuracy while enjoying a much lower time cost compared with FD. Besides, when m increases, the sketching accuracy of FFD slightly decreases, but its running speed increases especially for a larger ℓ. Data $\mathbf{A}_2$ differs $\mathbf{A}_1$ only on n, and we observe that a larger n improves the sketching accuracy of FFD. Finally, we decrease the dimension d in $\mathbf{A}_3$, and we find that the improvement of running speed for FFD becomes smaller than that in $\mathbf{A}_1$. All these observations are consistent with the proved results in Theorem 4.1.

All the results indicate that our proposed FFD has the potential to benefit the sketching-based online hashing.

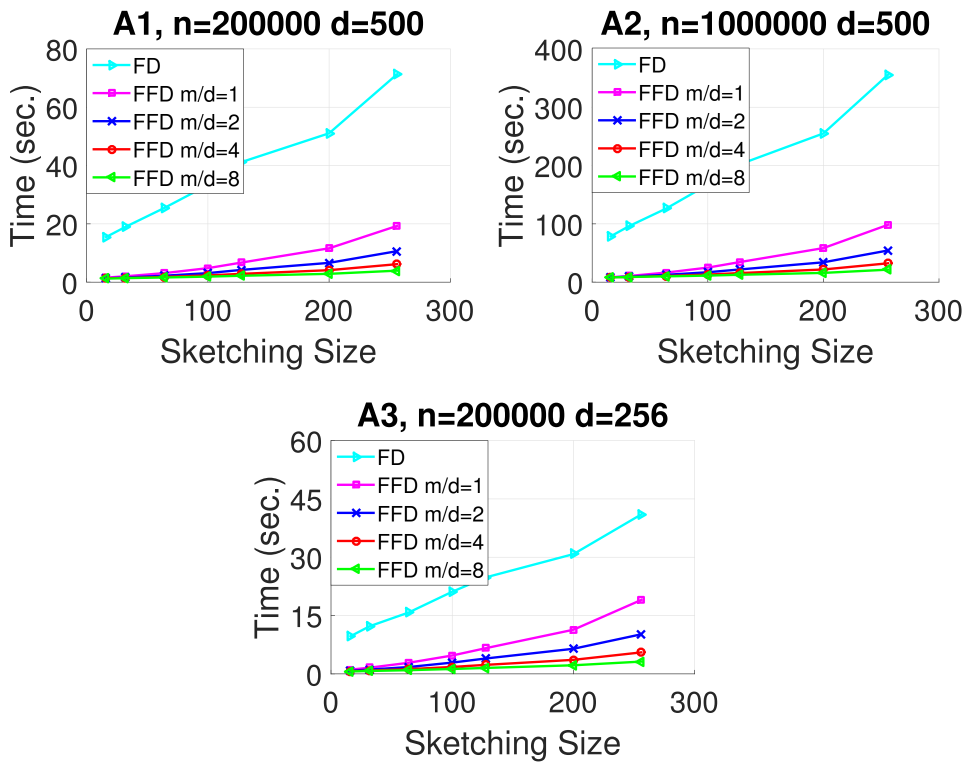

Figure 4.3: Time comparisons of sketching methods.

4.4.2 Comparisons with LSH and OSH

In this part, we compare against online unsupervised hashing methods including LSH [34] and OSH. We employ four datasets: CIFAR-10 [107], MNIST [33], GIST-1M [96], and FLICKR-25600 [199]. For CIFAR-10, it consists of 60,000 images in 10 classes with 6000 images per class. We extract 512-dimensional GIST descriptor to represent each image. MNIST contains 70,000 images with 784-dimensional features. GIST-1M consists of one million 960-dimensional GIST descriptors. The FLICKR-25600 dataset contains 100,000 images sampled from a noisy Internet image collection, and each image is represented by a 25,600-dimensional vector normalized to be of unit norm.

Following the common setting, we take the top 2% nearest neighbors in Euclidean space as the ground truth for all datasets.

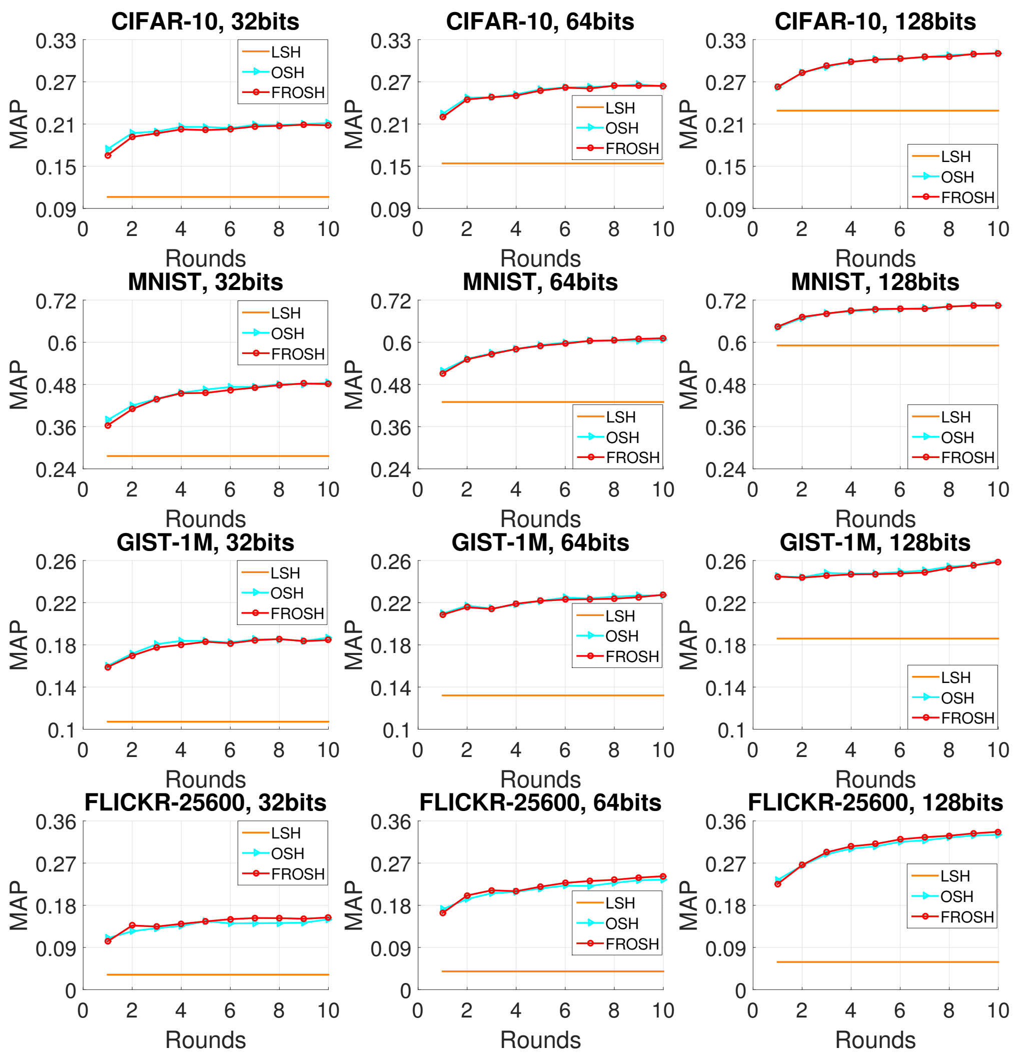

Figure 4.4: MAP at each round with 32, 64, and 128 bits.

In both OSH and FROSH, we set the sketching size $\ell = 2r$, where r is the code length assigned from $\{32, 64, 128\}$. We train both algorithms in a streaming fashion by evenly dividing each dataset into 10 parts, and evaluate the mean average precision (MAP) score after each round. We also empirically set $m = 4d$

in FROSH during the training. For LSH, it does not require training, and we simply report its MAP at the final round [111].

Figure 4.4 shows the MAP scores at different rounds on four datasets with 32, 64 and 128 bits codes. On all datasets, it is apparent that our proposed FROSH performs as accurately as OSH and outperforms LSH with a large margin. In addition, both OSH and FROSH stably improve the MAP scores when receiving more data, which demonstrates that a successful adaption to the data variations has been achieved. Table 4.3 provides the accumulated training time of OSH and FROSH with 32, 64, 128 bits codes, respectively. FROSH consistently has the lower training time for each comparison, which achieves about 10 $\sim$ 20 times speed-up than OSH. Thus, our method is highly efficient.

Table 4.2: Accumulated training time of OSH and FROSH after all rounds (in sec.).

Dataset	**Method**	32bits	64bits	128bits
CIFAR-10	OSH	7.78	11.88	22.09
	FROSH	**0.63**	**0.94**	**2.11**
MNIST	OSH	13.25	18.93	30.75
	FROSH	**1.17**	**1.49**	**2.56**
GIST-1M	OSH	228	331	520
	FROSH	**21**	**27**	**45**
FLICKR-25600	OSH	679	1283	2570
	FROSH	**72**	**92**	**134**

4.4.3 Comparisons with Batch Solutions

We also compare OSH and our FROSH against two leading methods SGH [97] and OCH [123]. Based on the literature [125, 97, 123], we know that SGH maintains a superior tradeoff between the learning accuracy and the scalability, and OCH is the state-of-the-art regarding the accuracy performance compared with the other unsupervised hashing methods such as SpH [185], AGH [126], IsoH [106], DGH [125], and OEH [124].

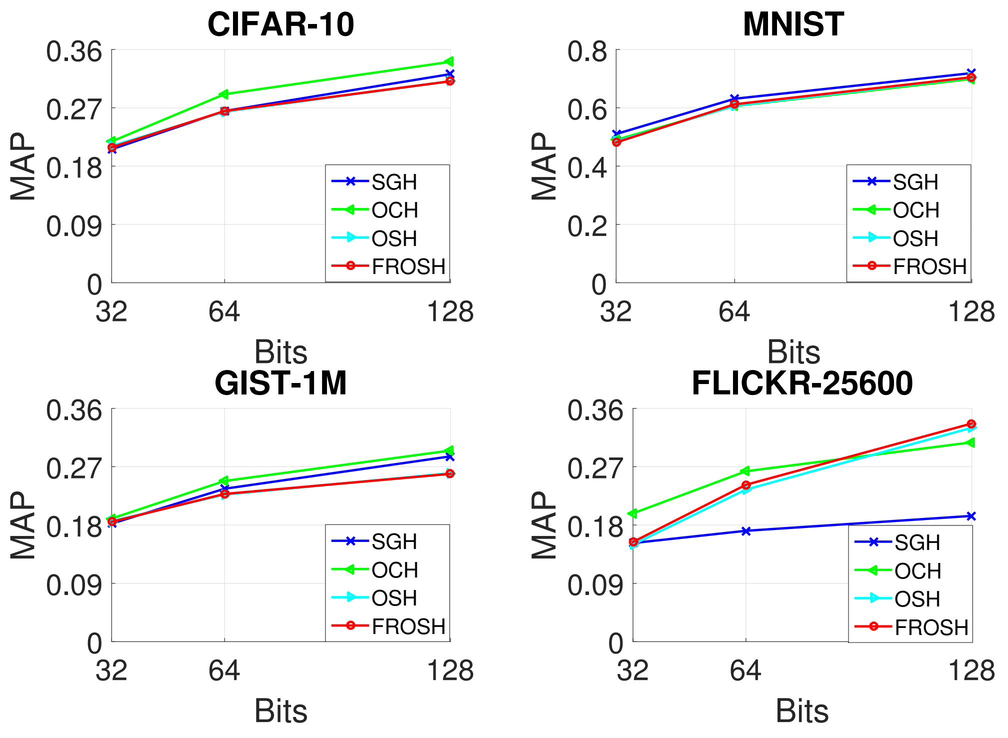

Figure 4.5: MAP comparisons at different code lengths.

In Figure 4.5, we clearly observe that FROSH achieves comparable or even better accuracy in comparison with the two leading batch solutions, which suggests that not only does the online sketching maintain sufficient information necessary for the hash function training but also it owns a good learning ability. In conclusion, considering the properties that FROSH can adapt the hash functions to the new coming data and enjoy superior training efficiency (single pass, lowest costs of space and time), our proposed method is more appropriate for hashing learning on the big streaming unlabeled data. Note that we defer the time comparisons in the appendix.

Besides, FROSH enjoys superior training efficiency, i.e., *single pass, lowest costs of space ($O(\ell d)$) and time*, while the unsupervised batch methods such as SGH and OCH require either $O(nd)$ space or multiple passes over the data to load all data into memory or both. Regarding the space cost, OSH and FROSH can throw away the used data and update only on the newly coming data via $O(\ell d)$ space while SGH and OCH should maintain a large extern storage to keep all observed data and the newly coming data. Moreover, given the 19GB data FLICKR-25600 that is merely a small subset of the entire FLICKR image collection, if avoiding multiple passes is required, both the SGH and OCH methods have to maintain the entire FLICKR-25600 data in the memory and keep the intermediate computational results (can be several times larger than FLICKR-25600 itself) in the memory as well, which is infeasible for common computers.

In Table 4.3, we also provide the empirical time comparisons because it is not clear enough to directly compare the associated time complexities when all methods have different parameters. Note that the released OCH codes have been highly optimized by the authors in contrast to the version used in its original paper [123]. We report the accumulated time after all rounds for OSH and FROSH and provide the time consumptions of training SGH and OCH only on all observed data. Overall, our FROSH offers about $10 \sim 70$ times speed-up than the other compared solutions. If considering the case where batch-learners should repeatedly do batch learning on both the newly coming data and the currently observed data, SGH and OCH would require significantly more training time than that in Table 4.3.

Table 4.3: Comparisons of the training time (in sec.). We report the accumulated time after all rounds for OSH and FROSH and provide the time consumptions of training SGH and OCH only on all observed data.

Dataset	**Method**	32bits	64bits	128bits
CIFAR-10	SGH	7.83	11.35	19.49
	OCH	26.89	26.95	27.49
	OSH	7.78	11.88	22.09
	FROSH	**0.63**	**0.94**	**2.11**
MNIST	SGH	10.47	14.59	23.47
	OCH	40.45	40.49	41.10
	OSH	13.25	18.93	30.75
	FROSH	**1.17**	**1.49**	**2.56**
GIST-1M	SGH	231	275	290
	OCH	1042	1089	1192
	OSH	228	331	520
	FROSH	**21**	**27**	**45**
FLICKR-25600	SGH	3032	3541	4903
	OCH	4981	5300	5441
	OSH	679	1283	2570
	FROSH	**72**	**92**	**134**

4.5 Conclusion

In this chapter, we propose an effective hashing method for streaming unlabeled data. Its basic idea is to reduce the sketching time that is the dominated cost in the OSH method.

The analysis shows better time efficiency with accuracy guaranteed. The simulations on both synthetic and real-world datasets support our theoretical findings and demonstrate the practicability of our method.

The sketching designed in this work also provides an insightful view for accelerating general streaming matrix multiplications, which then may be helpful for deriving an online version of OCH. Such extensions can be explored in the future work.

4.6 Proofs

4.6.1 Preliminaries

In our analysis, we turn to series of existing theoretical tools. We use the Matrix Bernstein inequality on the sum of zero-mean random matrices given as below.

Theorem 4.3 ([171]). *Let* $\{\mathbf{A}_i\}_{i=1}^L \in \mathbb{R}^{n\times d}$ *be independent random matrices with* $\mathbb{E}[\mathbf{A}_i] = \mathbf{0}^{n\times d}$ *and* $\|\mathbf{A}_i\|_2 \le R$ *for all* $i \in [L]$*. Define* $\sigma^2 = \max\{\|\sum_{i=1}^L \mathbb{E}\left[\mathbf{A}_i\mathbf{A}_i^T\right]\|_2, \|\sum_{i=1}^L \mathbb{E}\left[\mathbf{A}_i^T\mathbf{A}_i\right]\|_2\}$ *as the variance parameter. Then, for all* $\epsilon \ge 0$ *we have*

$$\mathbb{P}(\|\sum_{i=1}^L \mathbf{A}_i\|_2 \ge \epsilon) \le (d+n)\exp(\frac{-\epsilon^2/2}{\sigma^2 + R\epsilon/3}). \tag{4.7}$$

It is also helpful to provide the next result that characterizes the property of compressed data via SRHT matrix.

Theorem 4.4 ([128]). *Given* $\mathbf{A} \in \mathbb{R}^{m\times d}$*, let rank*$(\mathbf{A}) \le k \le \min(m,d)$ *and* $\mathbf{\Phi} \in \mathbb{R}^{q\times m}$ *be the SRHT matrix. Then, with probability at least* $1 - (\delta + \frac{m}{e^k})$ *we have*

$$(1-\Delta)\mathbf{A}^T\mathbf{A} \preceq \mathbf{A}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{A} \preceq (1+\Delta)\mathbf{A}^T\mathbf{A}, \tag{4.8}$$

where $\Delta = \Theta(\sqrt{\frac{k \log(2k/\delta)}{q}})$.

With Corollary 3 of [10], it is straightforward to have the next norm bound for compressed data vectors.

Lemma 4.1. *Give data matrix* $\mathbf{A} \in \mathbb{R}^{m \times d}$*, and the SRHT matrix* $\mathbf{\Phi} \in \mathbb{R}^{q \times m}$*. Then, with probability at least* $1 - \beta$*, we have*

$$\|\mathbf{\Phi a}_i\|_2 \leq \sqrt{2 \log(\frac{2md}{\beta})} \|\mathbf{a}_i\|_2, \textit{ for all} i \in [d]. \tag{4.9}$$

Before proceeding, we first give the following Lemma together with its proof.

Lemma 4.2. *Given data matrix* $\mathbf{X} \in \mathbb{R}^{m \times d}$*, and the scaled sampling matrix* $\mathbf{S} \in \mathbb{R}^{q \times m}$ *in SRHT. Then, we have*

$$\mathbb{E}[\mathbf{X}^T \mathbf{S}^T \mathbf{S} \mathbf{X}] = \mathbf{X}^T \mathbf{X}. \tag{4.10}$$

4.6.2 Proof of Lemma 4.2

Proof. It is related to sampling without replacement. According to the definition, each column vector in the scaled sampling matrix $\mathbf{S} \in \mathbb{R}^{q \times m}$ is sampled without replacement from $\{\mathbf{s}^i = \sqrt{\frac{m}{q}} \mathbf{e}_i^T\}_{i=1}^m \in \mathbb{R}^{1 \times m}$ uniformly. Note that

$$\mathbb{E}[\mathbf{X}^T \mathbf{S}^T \mathbf{S} \mathbf{X}] = \mathbb{E}[\mathbf{X}^T \sum_{i=1}^{q} \mathbf{s}^{i^T} \mathbf{s}^i \mathbf{X}] \tag{4.11}$$

$$= \mathbf{X}^T \sum_{i=1}^{q} \mathbb{E}[\mathbf{s}^{i^T} \mathbf{s}^i] \mathbf{X}. \tag{4.12}$$

Without loss of generality, we assume that we sample and determine the value of $\{\mathbf{s}^i\}_{i=1}^q$ in the order of the smallest to the largest index. Then, due to that

$$\begin{aligned}\mathbb{P}(\mathbf{s}^i = \sqrt{\frac{m}{q}}\mathbf{e}_k^T) = \mathbb{P}(\mathbf{s}^i = \sqrt{\frac{m}{q}}\mathbf{e}_k^T | \mathbf{s}^j \neq \sqrt{\frac{m}{q}}\mathbf{e}_k^T, \forall j \in [i-1]) \\ * \mathbb{P}(\mathbf{s}^j \neq \sqrt{\frac{m}{q}}\mathbf{e}_k^T, \forall j \in [i-1]) \quad (4.13)\\ = \frac{1}{m-(i-1)}\frac{m-(i-1)}{m-(i-2)}\cdots\frac{m-1}{m} \quad (4.14)\\ = \frac{1}{m}, \quad (4.15)\end{aligned}$$

we have

$$\begin{aligned}\mathbb{E}[{\mathbf{s}^i}^T\mathbf{s}^i] &= \sum_{k=1}^{m}\mathbb{P}(\mathbf{s}^i = \sqrt{\frac{m}{q}}\mathbf{e}_k^T)\frac{m}{q}\mathbf{e}_k\mathbf{e}_k^T \quad (4.16)\\ &= \sum_{k=1}^{m}\frac{1}{m}\frac{m}{q}\mathbf{e}_k\mathbf{e}_k^T \quad (4.17)\\ &= \frac{1}{q}\mathbf{I}_m. \quad (4.18)\end{aligned}$$

Thus, substituting Eq. (4.18) into Eq. (4.12) concludes the proof. □

We are now ready to prove our main results: Theorem 4.1, Theorem 4.2, and Corollary 4.1.

4.6.3 Proof of Theorem 4.1

Proof. To clarify, we let $q = \ell/2$, define p as the times that steps 7 to 12 in Algorithm 4.3 have been executed, and assume $p = \frac{n}{m}$ without loss of generality for the input $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2; \cdots; \mathbf{A}_p] \in \mathbb{R}^{n\times d}$ with $\{\mathbf{A}_t \in \mathbb{R}^{m\times d}\}_{t=1}^p$. By the triangle inequality, we have

$$\begin{aligned}\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2 \leq \|\mathbf{A}^T\mathbf{A} - \mathbf{C}^T\mathbf{C}\|_2 \\ + \|\mathbf{C}^T\mathbf{C} - \mathbf{B}^T\mathbf{B}\|_2, \quad (4.19)\end{aligned}$$

where $\mathbf{C} = [\mathbf{C}_1; \mathbf{C}_2; \cdots; \mathbf{C}_p] \in \mathbb{R}^{pq \times d}$ is from compressing each $\mathbf{A}_t$ by $\mathbf{\Phi}_t = \mathbf{S}_t\mathbf{H}\mathbf{D}_t$ in Algorithm 4.3 such that $\mathbf{C}_t = \mathbf{\Phi}_t\mathbf{A}_t$. Since $\mathbf{B}$ results from running standard FD on $\mathbf{C}$, then with probability at least $1 - p\beta$, we have

$$\begin{aligned} \|\mathbf{C}^T\mathbf{C} - \mathbf{B}^T\mathbf{B}\|_2 &\leq \frac{2}{\ell}\|\mathbf{C}\|_F^2 && (4.20) \\ &\leq \frac{4}{\ell}\log(\frac{2md}{\beta})\|\mathbf{A}\|_F^2, && (4.21) \end{aligned}$$

where Eq. (4.20) directly follows from the error bound of FD [118], and Eq. (4.21) holds by combing $\|\mathbf{C}\|_F^2 = \sum_{t=1}^p \|\mathbf{C}_t\|_F^2 = \sum_{t=1}^p \sum_{i=1}^d \|\mathbf{c}_{t,i}\|_2^2$ with Lemma 4.1 and the union bound.

We next bound $\|\mathbf{A}^T\mathbf{A} - \mathbf{C}^T\mathbf{C}\|_2$. Define $\mathbf{X}_t = \mathbf{H}\mathbf{D}_t\mathbf{A}_t$, then it follows that

$$\begin{aligned} \|\mathbf{A}^T\mathbf{A} - \mathbf{C}^T\mathbf{C}\|_2 &= \|\sum_{t=1}^p(\mathbf{A}_t^T\mathbf{A}_t - \mathbf{C}_t^T\mathbf{C}_t)\|_2 && (4.22) \\ &= \|\sum_{t=1}^p(\mathbf{A}_t^T\mathbf{A}_t - \mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t)\|_2. && (4.23) \end{aligned}$$

Let $\mathbf{Z}_t = \mathbf{A}_t^T\mathbf{A}_t - \mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t$, then obviously $\{\mathbf{Z}_t\}_{t=1}^p$ are independent random variables. By Lemma 4.2, we perform the expectation w.r.t. $\mathbf{S}_t$ and $\mathbf{D}_t$ to obtain that

$$\begin{aligned} \mathbb{E}[\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t] &= \mathbb{E}_{\mathbf{D}_t}\mathbb{E}_{\mathbf{S}_t}[\mathbf{X}_t^T\mathbf{S}_t^T\mathbf{S}_t\mathbf{X}_t|\mathbf{D}_t] && (4.24) \\ &= \mathbb{E}_{\mathbf{D}_t}[\mathbf{X}_t^T\mathbf{X}_t] \\ &= \mathbb{E}_{\mathbf{D}_t}[\mathbf{A}_t^T\mathbf{D}_t^T\mathbf{H}^T\mathbf{H}\mathbf{D}_t\mathbf{A}_t] \\ &= \mathbf{A}_t^T\mathbf{A}_t, \end{aligned}$$

where the second equality follows from Lemma 4.2 with $\mathbf{D}_t$ fixed, and the last equality holds because $\mathbf{H}$ and $\mathbf{D}_t$ are the unitary matrices. Thus, $\{\mathbf{Z}_t\}_{t=1}^p$ satisfy the setting of the Matrix Bernstein inequality in Theorem 4.3.

Hence, due to that $\|\mathbf{Z}_t\|_2 \leq \Delta_t \|\mathbf{A}_t^T \mathbf{A}_t\|_2 = \Delta_t \|\mathbf{A}_t\|_2^2$ resulted from Theorem 4.4, we first achieve

$$R = \max_{t\in[p]} \Delta_t \|\mathbf{A}_t\|_2^2 \tag{4.25}$$

with probability at least $1-(p\delta + \sum_{t=1}^{p} \frac{m}{e^{k_t}})$ after applying union bound, where $\Delta_t = \Theta(\sqrt{\frac{k_t \log(2k_t/\delta)}{q}})$ and $\text{rank}(\mathbf{A}_t) \leq k_t \leq \min(m, d)$.

Regarding σ^2, due to the symmetry of each matrix $\mathbf{Z}_t$, $\sigma^2 = \|\sum_{t=1}^{p} \mathbb{E}[(\mathbf{Z}_t)^2]\|_2$ holds. Next, we have

$$\begin{aligned}
\mathbf{0}^{d\times d} &\preceq \mathbb{E}[(\mathbf{Z}_t)^2] && (4.26)\\
&= \mathbb{E}[(\mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t)^2] - (\mathbf{A}_t^T \mathbf{A}_t)^2 && (4.27)\\
&\preceq \mathbb{E}[\|\mathbf{S}_t \mathbf{X}_t\|_2^2 \mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t] - (\mathbf{A}_t^T \mathbf{A}_t)^2 && (4.28)\\
&\preceq \mathbb{E}[(1+\Delta_t)\|\mathbf{A}_t\|_2^2 \mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t] - (\mathbf{A}_t^T \mathbf{A}_t)^2 && (4.29)\\
&= (1+\Delta_t)\|\mathbf{A}_t\|_2^2 \mathbf{A}_t^T \mathbf{A}_t - (\mathbf{A}_t^T \mathbf{A}_t)^2 && (4.30)
\end{aligned}$$

with probability at least $1-(\delta + \frac{m}{e^{k_t}})$, where Eqs. (4.27)(4.30) hold because $\mathbb{E}(\mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t) = \mathbf{A}_t^T \mathbf{A}_t$, Eq. (4.29) follows from Theorem 4.4, and Eq. (4.28) holds because of

$$\mathbf{0}^{d\times d} \preceq (\mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t)^2 \preceq \|\mathbf{S}_t \mathbf{X}_t\|_2^2 \mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t,$$

which results from that for any $\mathbf{y} \in \mathbb{R}^d$,

$$\begin{aligned}
\mathbf{y}^T (\mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t)^2 \mathbf{y} &= \|\mathbf{y}^T \mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t\|_2^2\\
&\leq \|\mathbf{y}^T \mathbf{X}_t^T \mathbf{S}_t^T\|_2^2 \|\mathbf{S}_t \mathbf{X}_t\|_2^2\\
&= \|\mathbf{S}_t \mathbf{X}_t\|_2^2 \mathbf{y}^T \mathbf{X}_t^T \mathbf{S}_t^T \mathbf{S}_t \mathbf{X}_t \mathbf{y}.
\end{aligned}$$

Then, we have

$$\|\sum_{t=1}^{p} \mathbb{E}[(\mathbf{Z}_t)^2]\|_2 \leq \sum_{t=1}^{p} \|\mathbb{E}[(\mathbf{Z}_t)^2]\|_2 \tag{4.31}$$

$$\leq \sum_{t=1}^{p} \left\|(1+\Delta_t)\|\mathbf{A}_t\|_2^2 \mathbf{A}_t^T \mathbf{A}_t - (\mathbf{A}_t^T \mathbf{A}_t)^2\right\|_2 \tag{4.32}$$

$$= \sum_{t=1}^{p} \left\|(1+\Delta_t)\|\mathbf{A}_t\|_2^2 \mathbf{U}_t \mathbf{\Sigma}_t^2 \mathbf{U}_t - \mathbf{U}_t \mathbf{\Sigma}_t^4 \mathbf{U}_t\right\|_2 \tag{4.33}$$

$$= \sum_{t=1}^{p} \left\|(1+\Delta_t)\|\mathbf{A}_t\|_2^2 \mathbf{\Sigma}_t^2 - \mathbf{\Sigma}_t^4\right\|_2 \tag{4.34}$$

$$= \sum_{t=1}^{p} \max_{j\in[d]} |(1+\Delta_t)\sigma_{t1}^2 \sigma_{tj}^2 - \sigma_{tj}^4| \tag{4.35}$$

$$\leq \sum_{t=1}^{p} (1+\Delta_t)\sigma_{t1}^4 \tag{4.36}$$

$$= \sum_{t=1}^{p} (1+\Delta_t)\|\mathbf{A}_t\|_2^4 \tag{4.37}$$

$$\leq \max_{t\in[p]} p(1+\Delta_t)\|\mathbf{A}_t\|_2^4. \tag{4.38}$$

where Eq. (4.32) establishes due to Eq. (4.30), and $\mathbf{U}_t$ in Eq. (4.33) is from the SVD of $\mathbf{A}_t$ with $\mathbf{A}_t = \mathbf{U}_t \mathbf{\Sigma}_t \mathbf{V}_t^T$ and the eigenvalues $\sigma_{tj} \triangleq \sigma_{t,jj}$ listed in the descending order in $\mathbf{\Sigma}_t$. By Theorem 4.3, we have

$$\mathbb{P}(\|\sum_{t=1}^{p} \mathbf{Z}_t\|_2 \geq \epsilon) \leq 2d \exp(\frac{-\epsilon^2/2}{\sigma^2 + R\epsilon/3}). \tag{4.39}$$

Denote the RHS of Eq. (4.39) by δ, we then obtain that

$$\epsilon = \log(\frac{2d}{\delta})\Big(\frac{R}{3} + \sqrt{(\frac{R}{3})^2 + \frac{2\sigma^2}{\log(2d/\delta)}}\Big) \tag{4.40}$$

$$\leq \log(\frac{2d}{\delta})\frac{2R}{3} + \sqrt{2\sigma^2 \log(\frac{2d}{\delta})} \tag{4.41}$$

$$\leq \max_{t\in[p]} \widetilde{O}\Big(\Delta_t\|\mathbf{A}_t\|_2^2\Big) + \max_{t\in[p]} \widetilde{O}\Big(\sqrt{p(1+\Delta_t)}\|\mathbf{A}_t\|_2^2\Big) \tag{4.42}$$

$$\leq \max_{t\in[p]} \widetilde{O}\Big((\sqrt{\frac{k}{\ell p^2}} + \sqrt{\frac{1+\sqrt{k/\ell}}{p}})\frac{\|\mathbf{A}_t\|_2^2}{\|\mathbf{A}_t\|_F^2}\Big)\|\mathbf{A}\|_F^2 \tag{4.43}$$

$$\leq \widetilde{O}\Big((\sqrt{\frac{k}{\ell p^2}} + \sqrt{\frac{1+\sqrt{k/\ell}}{p}})\Big)\|\mathbf{A}\|_F^2. \tag{4.44}$$

To derive Eq. (4.43) from Eq. (4.42), we first substitute $\Delta_t = \Theta(\sqrt{\frac{k_t\log(2k_t/\delta)}{q}})$ into Eq. (4.42) and set $k = k_t = \min(m, d)$, which allows Eq. (4.42) to become the maximum of the sum of two functions. Then, we leverage the definition $q = \ell/2$ and apply a common practical assumption of that $p\lambda_1 \leq \|\mathbf{A}\|_F^2 = \sum_{t=1}^p \|\mathbf{A}_t\|_F^2 \leq p\lambda_2$ with each $\|\mathbf{A}_t\|_F^2$ bounded between λ_1 and λ_2 that are very close to each other [10, 148].

Combing Eq. (4.44) with Eqs. (4.19)(4.21) by the union bound achieves the desired result with probability at least $1 - p\beta - (2p+1)\delta - 2p\frac{m}{e^k}$.

The computational analysis is straightforward based on Sections 4.2.2 and 4.3.2. □

4.6.4 Proof of Theorem 4.2

Proof. This proof directly follows from that in Section 4.2.1 and OSH [111] since FROSH differs OSH merely in the sketching techniques. Then, leveraging Theorem 4.1 immediately yields the desired results.

Let $\boldsymbol{\mu}_j \in \mathbb{R}^{1\times d}$, $\hat{\boldsymbol{\mu}}_t \in \mathbb{R}^{1\times d}$ and $\boldsymbol{\mu} \in \mathbb{R}^{1\times d}$ be the mean vector of $\mathbf{A}_j \in \mathbb{R}^{h_j\times d}$, $\{\mathbf{A}_j\}_{j=1}^t$ and $\mathbf{A} = \{\mathbf{A}_j\}_{j=1}^s \in \mathbb{R}^{n\times d}$ respectively, where $t \leq s$. Obviously, $\hat{\boldsymbol{\mu}}_s = \boldsymbol{\mu}$ holds.

Then, the data centering procedure in FROSH (and OSH) always ensures that $\sum_{j=1}^t(\mathbf{A}_j - \boldsymbol{\mu}_j)^T(\mathbf{A}_j - \boldsymbol{\mu}_j) = (\{\mathbf{A}_j\}_{j=1}^t -$

$\hat{\boldsymbol{\mu}}_t)^T(\{\mathbf{A}_j\}_{j=1}^t - \hat{\boldsymbol{\mu}}_t)$, where $\mathbf{A}_j - \boldsymbol{\mu}_j$ means subtracting each row vector of $\mathbf{A}_j$ by $\boldsymbol{\mu}_j$, and $\{\mathbf{A}_j\}_{j=1}^t$ can be regarded as a $(\sum_{j=1}^T h_j)\times d$ matrix by vertically stacking all $\mathbf{A}_j$.

In addition, setting $t = s$ and using $\|\mathbf{A} - \boldsymbol{\mu}\|_F^2 = \mathrm{Tr}((\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu}))$ immediately yields the desired result via replacing the $\mathbf{A}$ in the sketching bound of FFD Eq. (4.4) in Theorem 4.1 by $\mathbf{A} - \boldsymbol{\mu}$.

The computational analysis also follows from OSH, as the difference only lies in that between FFD and FD. □

4.6.5 Proof of Corollary 4.1

Proof. The proof follows by combining our proposed Theorem 4.2 and Section 1.4 of [118]. Since $\mathbf{W}_{\mathbf{B}}^T \in \mathbb{R}^{r\times d}$ contains the top r right singular vectors of $\mathbf{B}^{\ell\times d}$, we have $\mathbf{W}_{\mathbf{B}}\mathbf{W}_{\mathbf{B}}^T \in \mathbb{R}^{d\times d}$ as the projection matrix of $\mathbf{B}^{\ell\times d}$. Via Lemma 4 in [55], we have

$$\begin{aligned}\|(\mathbf{A} - \boldsymbol{\mu}) - (\mathbf{A} - \boldsymbol{\mu})\mathbf{W}_{\mathbf{B}}\mathbf{W}_{\mathbf{B}}^T\|_2^2 \\ \le \sigma_{r+1}^2 + 2\|(\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu}) - \mathbf{B}^T\mathbf{B}\|_2,\end{aligned} \tag{4.45}$$

where σ_i is the i-th largest singular value of $(\mathbf{A} - \boldsymbol{\mu})$.

For simplicity, when $m = \Theta(d)$ and $n = \Omega(\ell^{3/2}d^{3/2})$, the error bound of Eq. (5) in Theorem 4.2 will become $\widetilde{O}(\frac{1}{\ell}\|(\mathbf{A} - \boldsymbol{\mu})\|_F^2)$, which is then incorporated into Eq. (4.45) to get that

$$\begin{aligned}\|(\mathbf{A} - \boldsymbol{\mu}) - (\mathbf{A} - \boldsymbol{\mu})\mathbf{W}_{\mathbf{B}}\mathbf{W}_{\mathbf{B}}^T\|_2^2 \\ \le \sigma_{r+1}^2 + \widetilde{O}(\frac{1}{\ell}\|(\mathbf{A} - \boldsymbol{\mu})\|_F^2).\end{aligned} \tag{4.46}$$

Let $h = \|(\mathbf{A} - \boldsymbol{\mu})\|_F^2/\|(\mathbf{A} - \boldsymbol{\mu})\|_2^2$ be the numeric rank of $(\mathbf{A} - \boldsymbol{\mu}) \in \mathbb{R}^{n\times d}$, which could be much smaller than d for a low-rank matrix $(\mathbf{A} - \boldsymbol{\mu}) \in \mathbb{R}^{n\times d}$ with $d < n$. If $\ell = \Omega(\frac{h\sigma_1^2}{\epsilon\sigma_{r+1}^2})$, then from Eq. (4.46) we have

$$\begin{aligned}\|(\mathbf{A}-\boldsymbol{\mu})-(\mathbf{A}-\boldsymbol{\mu})\mathbf{W}_{\mathbf{B}}\mathbf{W}_{\mathbf{B}}^T\|_2^2 &\le (1+\epsilon)\sigma_{r+1}^2\\ &= (1+\epsilon)\|(\mathbf{A}-\boldsymbol{\mu})-(\mathbf{A}-\boldsymbol{\mu})\mathbf{W}\mathbf{W}^T\|_2^2, \qquad (4.47)\end{aligned}$$

where $\sigma_1^2 = \|(\mathbf{A}-\boldsymbol{\mu})\|_2^2$ and $\sigma_{r+1}^2 = \|(\mathbf{A}-\boldsymbol{\mu})-(\mathbf{A}-\boldsymbol{\mu})\mathbf{W}\mathbf{W}^T\|_2^2$ according to the definition. □

□ **End of chapter.**

Chapter 5

Toward Efficient and Accurate Covariance Matrix Estimation on Compressed Data

Estimating covariance matrices is a fundamental technique in various domains, most notably in machine learning and signal processing. To tackle the challenges of extensive communication costs, large storage capacity requirements, and high processing time complexity when handling massive high-dimensional and distributed data, we propose an efficient and accurate covariance matrix estimation method via data compression. In contrast to previous data-oblivious compression schemes, we leverage a data-aware weighted sampling method to construct low-dimensional data for such estimation. We rigorously prove that our proposed estimator is unbiased and requires smaller data to achieve the same accuracy with specially designed sampling distributions. Besides, we depict that the computational procedures in our algorithm are efficient. All achievements imply an improved tradeoff between the estimation accuracy and computational costs. Finally, the extensive experiments on synthetic and real-world datasets validate the superior property of our method and illustrate that it significantly outperforms the state-of-the-art algorithms.

5.1 Introduction

Covariance matrices play a fundamental role in machine learning and statistics owing to their capability to retain the second-order information of data samples [67]. For example, Principal Component Analysis (PCA) along with its extensions [205], Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA) [14] are powerful for dimension reduction and denoising, which require the estimation of a covariance matrix from a given collection of data points. Other prominent examples include Generalized Least Squares (GLS) regression that requires the estimation of the noise covariance matrix [103], Independent Component Analysis (ICA) that relies on pre-whitening based on the covariance matrix [93], and Generalized Method of Moments (GMM) [84] that improves the effectiveness by a precise covariance matrix.

Many practical applications also rely on covariance matrix directly [17]. In biology, gene relevance networks and gene association networks are straightforwardly inferred from the covariance matrix [27, 158]. In modern wireless communications, protocols optimize the bandwidth based on covariance estimates [172]. In array signal processing, the capon beamformer linearly combines the sensors to minimize the noise in the signal, which is closely related to the portfolio optimization on covariance matrices [2]. For policy learning in the field of robotics, it requires reliable estimates of the covariance matrix between policy parameters [51].

Calculation of a covariance matrix usually requires enormous computational resources in the form of communication and storage because large and high-dimensional data are now routinely gathered at an exploding rate from many distributed remote sites, such as sensor networks, surveillance, and distributed databases [78, 86, 164]. In particular, high communication cost of transmitting the distributed data from the remote sites to the fusion center (i.e., a destination to conduct complex data analysis tasks) will require tremendous bandwidth and power consumption [1, 168]. Formally, given a data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ with d features and n instances collected from the remote sites, the covariance matrix is computed in the fusion center by $\mathbf{C} \triangleq \frac{1}{n}\mathbf{X}\mathbf{X}^T - \bar{\mathbf{x}}\bar{\mathbf{x}}^T$, where $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \in \mathbb{R}^d$ [67]. For simplicity of discussion, we temporarily assume the empirical mean is zero, i.e., $\bar{\mathbf{x}} = \mathbf{0}$. The covariance matrix can be written as $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ consequently [16]. Then, it takes $O(nd)$ communication burden to transmit data from numerous remote sites to the fusion center to form the full data set $\mathbf{X}$, $O(nd)$ storage in total to store $\mathbf{X}$ in remote sites, and $O(nd+d^2)$ storage with $O(nd^2)$ time to calculate $\mathbf{C}$ in the fusion center. When $n, d \gg 1$, the overall cost is prohibitively expensive for practical scenarios like wireless sensors which have narrow transmission bandwidth, limited storage, and low power supply.

To tackle such computational challenges, compressed data can be leveraged to estimate the covariance matrix, which essentially has roots in compressed sensing. One solution is to process each data point by multiplying it with a single projection matrix $\mathbf{S} \in \mathbb{R}^{d\times m}$ whose entry follows the Gaussian distribution $\mathcal{N}(0, \frac{1}{m})$ [130]. Thus, storing $\mathbf{S}^T\mathbf{X}$ and the estimated covariance matrix requires $O(mn+d^2)$ space in total, sending $\mathbf{S}^T\mathbf{X}$ to the fusion center incurs a $O(mn)$ communication cost, and calculating $\mathbf{S}^T\mathbf{X}$ and the covariance matrix estimator $\frac{1}{n}\mathbf{S}\mathbf{S}^T\mathbf{X}\mathbf{X}^T\mathbf{S}\mathbf{S}^T$ takes $O(mdn + m^2n + m^2d + md^2)$ time. This method substantially reduces all computational costs if $m \ll n, d$. Note that synchronizing only a seed between remote sites and the fusion center allows pseudo-random number generators to reconstruct an identical $\mathbf{S}$, which avoids sending $\mathbf{S}$ directly and imposes a negligible computational burden.

However, the example solution has two critical drawbacks. The first is that the operations on the Gaussian matrix is inefficient. One could use a sparse projection matrix [113], structured matrix [7] or sampling matrix [58] to achieve a better tradeoff between computational cost and estimation precision. The second problem is that applying a single projection matrix to all data points cannot consistently estimate the covariance matrix, i.e., the estimator cannot converge to the actual covariance matrix even if the sample size n grows to infinity with d fixed. This issue is demonstrated both theoretically and empirically in [16] and also briefly described in [10, 11, 74].

In this chapter, we thus adopt n distinct projection matrices for n data vectors [9, 10, 11, 16] to achieve consistent covariance matrix estimation, and construct a specific sampling matrix to increase both its efficiency and accuracy. On the whole, we do not make statistical assumptions on the distributed data $\mathbf{X} \in \mathbb{R}^{d\times n}$ with $n, d \gg 1$, nor do we impose structural assumptions on the covariance matrix $\mathbf{C}$ such as being low-rank or sparse. Our goal is to compress data and recover $\mathbf{C}$ efficiently and accurately, and the contributions in our work are summarized as follows:

- First, in contrast to all existing methods [9, 10, 11, 16] that are based on data-oblivious projection matrices, we propose to estimate the covariance matrix based on the data compressed by a weighted sampling scheme. This strategy is data-aware with a capacity to explore the most important entries. Hence, we require considerably fewer entries to achieve an equal estimation accuracy.

- Second, we provide error analysis for the derived unbiased covariance estimator, which rigorously demonstrates that our method can compress data to a much smaller volume than other methods. The proofs also indicate our probability distribution is specifically designed to render a covariance matrix estimation based on the compressed data as accurate as possible.

- Third, we specify our method by an efficient algorithm whose computational complexity is superior to other methods. By additionally considering the best tradeoff between the estimation accuracy and the compression ratio, our algorithm ultimately incurs a significantly lower computational cost than the other methods.

- Finally, we validate our method on both synthetic and real-world datasets, which demonstrates a better performance than the other methods.

The remainder of this chapter is organized as follows. In Section 5.2, we review the prior work. In Section 5.3, we present our method along with theoretical analysis and emphasize its achievements. In Section 5.4, we provide extensive empirical results, and in Section 5.5 we conclude the whole work.

5.2 Related Work

There have been several investigations of ways to achieve accurate covariance matrix estimation from the low-dimensional compressed observations constructed by applying a distinct projection matrix $\{\mathbf{S}_i\}_{i=1}^n \in \mathbb{R}^{d\times m}$ to each data vector $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$. The work of [152] adopts a Gaussian matrix to compress data via $\mathbf{S}_i^T\mathbf{x}_i$, and recovers them by $\mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}(\mathbf{S}_i^T\mathbf{x}_i)$. Because $\mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T$ is a strictly m-dimensional orthogonal projection drawn uniformly at random, it can capture the information of all entries in each data vector uniformly and substantively. Then, $\frac{1}{n}\sum_{i=1}^n \mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T$ up to a known scaling factor is expected to constitute accurate and consistent covariance matrix estimation. This estimator can be modified to an unbiased one, and its error analysis is thoroughly provided in [16]. However, a Gaussian matrix is dense and unstructured, which imposes an extra computational burden. Also, many matrix inversions take a considerable amount of time, and the whole square matrix has to be loaded into the memory. Biased estimator $\frac{1}{n}\sum_{i=1}^n \mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$ is thus proposed in [11] to improve the efficiency by avoiding matrix inversions and assigning $\mathbf{S}_i$ to be a sparse matrix. This method is less accurate because $\mathbf{S}_i\mathbf{S}_i^T$ approximates only an m-dimensional random orthogonal projection. Its another disadvantage is that the result only holds for data samples under statistical assumptions. Based on [11], another study proposes an unbiased estimator [9], but it still adopts an unstructured sparse matrix that is insufficiently computation-efficient and fails to provide the error bounds to characterize the estimation error versus the compression ratio. Recently, sampling matrices $\mathbf{S}_i \in \mathbb{R}^{d\times m}$

constructed via uniform sampling *without replacement* have been employed [10]. This approach is efficient, but it only results in poor accuracy if data are compressed directly by $\mathbf{S}_i^T \mathbf{x}_i \in \mathbb{R}^d$ because $\mathbf{S}_i \mathbf{S}_i^T$ is an m-dimensional orthogonal projection drawn only from d deterministic orthogonal spaces/coordinates, and the $d - m$ entries of each vector are removed. To avoid sacrificing much accuracy, use of the computationally efficient Hadamard matrix [170] before sampling has also been proposed in [10]. It flattens out whole entries, particularly those with large magnitudes, to all coordinates to ensure that poor uniform sampling with a small sampling size still obtains some information among all entries. However, the Hadamard matrix involves deterministic orthogonal projection and is unable to capture the information uniformly in all coordinates of each vector, which results in the need for numerous samples to achieve sufficient accuracy. [10] constitutes the current state of the art in the tradeoff between the estimation accuracy and computational efficiency. Throughout this chapter, we group the foregoing representative methods into *Gauss-Inverse* [16, 152], *Sparse* [9, 11], and *UniSample-HD* [10], and the unbiased estimators produced by these methods are adopted in the subsequent theoretical and empirical comparisons.

A number of other methods have been proposed to recover covariance matrix from compressed data [21, 28, 38, 45]. These methods are only applicable to low-rank, sparse, or statistically-assumed covariance matrices.

Interesting work has also been done in the area of low-rank matrix approximation via randomized techniques. In addition to simply embedding the data $\mathbf{X}$ into space spanned by a single random projection matrix $\mathbf{S}$, a representative study [79] improves approximation accuracy by replacing the random projection matrix $\mathbf{S}$ with a low-dimensional data-aware matrix $\mathbf{XS}'$, where $\mathbf{S}'$ is a random projection matrix. However, $\mathbf{X}$ has to be low-rank, and computing $\mathbf{XS}'$ requires one extra pass through all entries in $\mathbf{X}$. It is not suitable for our settings, where we do not impose structural assumptions on the covariance matrix, nor do we fully observe all data. Moreover, [16] demonstrates both theoretically and empirically that a single projection matrix for all data points cannot consistently and accurately estimate the covariance matrix. The problem also exist in [138, 190] aiming for a fast approximation of matrix products in a single pass, which only results in an inconsistent covariance matrix estimation and suits the low-rank case.

Among randomized techniques, it is also worth briefly discussing sampling approaches in matrix approximation. Literature in [56, 89, 146, 188] proposes to leverage column sampling in which the sampling probabilities in the sampling matrix are either the column norms or leverage scores. Other work [4, 5, 188] performs element-wise sampling on the entire matrix based on the relative magnitudes over all data entries. These researches employ different sampling distributions to sample entries in a matrix. However, they have to observe all data fully to calculate the sampling distributions, which also requires one or more extra passes. In addition, their sampling probabilities are designed for matrix approximation, which cannot be trivially extended to covariance matrix estimation because the exact covariance matrix in our setting cannot be calculated in advance. Note that although the uniform sampling in matrix approximation is a simple one-pass algorithm, it performs poorly on many problems because usually there exists structural non-uniformity in the data which has been verified in [10].

5.3 Efficient and Accurate Covariance Estimation

In this section, we first introduce the definition and background to our overall work. We then justify and present our method of data compression and covariance matrix estimation, followed by the primary results and analysis.

5.3.1 Method and Algorithm

As discussed previously, *Gauss-Inverse* [16, 152] and *Sparse* [9, 11] suffer from deficiencies in either computational efficiency or estimation accuracy, whereas *UniSample-HD* [10] is less accurate but offers a good tradeoff between estimation accuracy and computational efficiency. We thus propose the adoption of weighted sampling matrices $\{\mathbf{S}_i\}_{i=1}^n \in \mathbb{R}^{d\times m}$ to compress data via $\mathbf{S}_i^T\mathbf{x}_i$ and then back-project the compressed data into the original space via $\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i$. The recovered data is then used for covariance matrix estimation as shown in Eq. (5.1). Hence, a high computational efficiency is maintained. Although $\mathbf{S}_i$ removes at least $d-m$ entries from the i-th vector, the remainders can be the most informative and are retained. With the carefully designed sampling probabilities, our unbiased estimator $\mathbf{C}_e$ performs as accurately as or more accurately than its counterparts asymptotically in terms of matrix spectral norm $\|\mathbf{C}_e - \mathbf{C}\|_2$. Note we have not quantified the error in any other entry-wise norm (e.g., the Frobenius norm) that could be uninformative on the quality of the approximate invariant subspace and unstable regarding the additive random error [4, 9, 73].

Algorithm 5.1 The proposed algorithm.

```
Input:
    Data $\mathbf{X} \in \mathbb{R}^{d\times n}$, sampling size $m$, and $0 < \alpha < 1$.
Initialization:
    Estimated covariance matrix $\mathbf{C}_e \in \mathbb{R}^{d\times d}$.
 1: Initialize $\mathbf{Y} \in \mathbb{R}^{m\times n}$, $\mathbf{T} \in \mathbb{R}^{m\times n}$, $\mathbf{v} \in \mathbb{R}^{n}$, and $\mathbf{w} \in \mathbb{R}^{n}$ with $\mathbf{0}$.
 2: for all $i \in [n]$ do
 3:     Load $\mathbf{x}_i$ into memory, let $v_i = \|\mathbf{x}_i\|_1 = \sum_{k=1}^{d} |x_{ki}|$ and $w_i = \|\mathbf{x}_i\|_2^2 = \sum_{k=1}^{d} x_{ki}^2$
 4:     for all $j \in [m]$ do
 5:         Pick $t_{ji} \in [d]$ with $p_{ki} \equiv \mathbb{P}(t_{ji} = k) = \alpha\frac{|x_{ki}|}{v_i} + (1-\alpha)\frac{x_{ki}^2}{w_i}$, and let $y_{ji} = x_{t_{ji}i}$
 6:     end for
 7: end for
 8: Pass the compressed data $\mathbf{Y}$, sampling indices $\mathbf{T}$, $\mathbf{v}$, $\mathbf{w}$, and $\alpha$ to the fusion center.
 9: for all $i \in [n]$ do
10:     Initialize $\mathbf{S}_i \in \mathbb{R}^{d\times m}$ and $\mathbf{P} \in \mathbb{R}^{d\times n}$ with $\mathbf{0}$
11:     for all $j \in [m]$ do
12:         Let $p_{t_{ji}i} = \alpha\frac{|y_{ji}|}{v_i} + (1-\alpha)\frac{y_{ji}^2}{w_i}$, and $s_{t_{ji}j,i} = \frac{1}{\sqrt{mp_{t_{ji}i}}}$
13:     end for
14: end for
15: Compute $\mathbf{C}_e$ as defined in Eq. (5.1) by using $\{\mathbf{S}_i\}_{i=1}^{n}$, $\mathbf{T}$, $\mathbf{P}$, and $\mathbf{Y}$.
```

We here summarize our method in Algorithm 5.1. In a nutshell, we employ a weighted sampling that is able to explore the most important entries to reduce estimation error $\|\mathbf{C}_e - \mathbf{C}\|_2$. Steps 1 to 7 in our proposed algorithm show how to compress distributed data in many remote sites. In step 5, each entry is retained with probability proportional to the combination of its relative absolute value and square value, and such sampling probability is designed to make $\|\mathbf{C}_e - \mathbf{C}\|_2$ as small as possible. Step 8 shows the communication procedure, and steps 9 to 15 reveal how to construct an unbiased covariance matrix estimator in the fusion center from compressed data. In many computing cases, it is possible to manipulate vectors of length $O(d)$ in memory, and thus when compressing data via weighted sampling, only one pass is required to move data from the external space to memory. Hence, our algorithm is also applicable to streaming data. For a covariance matrix defined as $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T - \bar{\mathbf{x}}\bar{\mathbf{x}}^T$, we can exactly calculate $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ in the fusion center via $\bar{\mathbf{x}} = \frac{1}{n}\sum_{j=1}^{g}\mathbf{u}_j$, where $\{\mathbf{x}_i\}_{i=1}^n$ are from $g \ll n$ remote sites, and $\mathbf{u}_j \in \mathbb{R}^d$ is the summation of all data vectors in the j-th remote site before being compressed. Doing so makes no deviation on the following error analysis and imposes only a negligible computational burden.

5.3.2 Primary Provable Results

In this part, we introduce the proposed covariance matrix estimator. In Algorithm 5.1, we employ $\mathbf{Y}$, $\mathbf{T}$, $\mathbf{v}$, and $\mathbf{w}$ to calculate $\{\mathbf{S}_i\}_{i=1}^n$. It can be verified that using only $\{\mathbf{S}_i\}_{i=1}^n$ and $\mathbf{Y}$ is able to obtain $\{\mathbf{S}_i^T\mathbf{x}_i\}_{i=1}^n$. Thus, we describe our estimator via $\{\mathbf{S}_i\}_{i=1}^n$ and $\{\mathbf{S}_i^T\mathbf{x}_i\}_{i=1}^n$ in the following theorem, which shows our estimator is unbiased.

Theorem 5.1. *Assume $\mathbf{X} \in \mathbb{R}^{d\times n}$ and the sampling size $2 \leq m < d$. Sample m entries from each $\mathbf{x}_i \in \mathbb{R}^d$ with replacement by running Algorithm 5.1. Let $\{p_{ki}\}_{k=1}^d$ and $\mathbf{S}_i \in \mathbb{R}^{d\times m}$ denote the sampling probabilities and sampling matrix, respectively. Then, the unbiased estimator for the target covariance matrix $\mathbf{C} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ can be recovered as*

$$\mathbf{C}_e = \widehat{\mathbf{C}}_1 - \widehat{\mathbf{C}}_2, \tag{5.1}$$

where $\mathbb{E}[\mathbf{C}_e] = \mathbf{C}$, $\widehat{\mathbf{C}}_1 = \frac{m}{nm-n}\sum_{i=1}^n \mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$, *and* $\widehat{\mathbf{C}}_2 = \frac{m}{nm-n}\sum_{i=1}^n \mathbb{D}(\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T)\mathbb{D}(\mathbf{b}_i)$ *with* $b_{ki} = \frac{1}{1+(m-1)p_{ki}}$.

Note that at most m entries in each $\mathbf{b}_i$ have to be calculated because each $\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$ has at most m non-zero entries in its diagonal. Now, having achieved the above unbiased estimator $\mathbf{C}_e$, we analyze its properties. We precisely upper bound the estimation error for the original estimator $\mathbf{C}$ in the matrix spectral norm.

Theorem 5.2. *Given* $\mathbf{X} \in \mathbb{R}^{d\times n}$ *and the sampling size* $2 \le m < d$, *let* $\mathbf{C}$ *and* $\mathbf{C}_e$ *be defined as in Theorem 5.1. If the sampling probabilities satisfy* $p_{ki} = \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$ *with* $0 < \alpha < 1$ *for all* $k \in [d]$ *and* $i \in [n]$, *then with probability at least* $1-\eta-\delta$,

$$\|\mathbf{C}_e - \mathbf{C}\|_2 \le \log(\frac{2d}{\delta})\frac{2R}{3} + \sqrt{2\sigma^2\log(\frac{2d}{\delta})}, \qquad (5.2)$$

where $R = \max_{i\in[n]}\left[\frac{7\|\mathbf{x}_i\|_2^2}{n} + \log^2(\frac{2nd}{\eta})\frac{14\|\mathbf{x}_i\|_1^2}{nm\alpha^2}\right]$, *and* $\sigma^2 = \sum_{i=1}^n \Big[\frac{8\|\mathbf{x}_i\|_2^4}{n^2m^2(1-\alpha)^2} + \frac{4\|\mathbf{x}_i\|_1^2\|\mathbf{x}_i\|_2^2}{n^2m^3\alpha^2(1-\alpha)} + \frac{9\|\mathbf{x}_i\|_2^4}{n^2m(1-\alpha)} + \frac{2\|\mathbf{x}_i\|_2^2\|\mathbf{x}_i\|_1^2}{n^2m^2\alpha(1-\alpha)}\Big] + \|\sum_{i=1}^n \frac{\|\mathbf{x}_i\|_1^2\mathbf{x}_i\mathbf{x}_i^2}{n^2m\alpha}\|_2$.

A large R and σ^2 work against the accuracy of $\mathbf{C}_e$. Accordingly, our sampling probabilities are designed to make R and σ^2 as small as possible to improve the accuracy. In the proof of Theorem 5.2, we also show that the selection of $q = 1, 2$ in $\frac{|x_{ki}|^q}{\sum_{k=1}^d |x_{ki}|^q}$ used for constructing the sampling probability $p_{ki} = \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$ is necessary and sufficient to make the error bound considerably tight.

Furthermore, α balances the performance by ℓ_1-norm based sampling $\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}$ and ℓ_2-norm based sampling $\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$. ℓ_2 sampling penalizes small entries more than ℓ_1 sampling. Hence ℓ_2 sampling is more likely to select larger entries to decrease error.

However, as seen from the proof in the *appendix*, different from ℓ_1 sampling, ℓ_2 sampling is unstable and sensitive to small entries, and it can make estimation error incredibly high if extremely small entries are picked. Hence, if α varies from 1 to 0, the estimation error will decrease and then increase, which is also empirically verified in the *appendix*.

The error bound in Theorem 5.2 involves many data-dependent quantities, whereas our primary interest lies in studying the tradeoff between the computational efficiency and estimation accuracy by employing weighted sampling to compress data and estimate covariance matrix. To clarify, we modify Theorem 5.2 and make the bound explicitly dependent on n, d, and m with the constraint $2 \leq m < d$.

Corollary 5.1. *Given $\mathbf{X} \in \mathbb{R}^{d\times n}$ and the sampling size $2 \leq m < d$, let $\mathbf{C}$ and $\mathbf{C}_e$ be created by Algorithm 5.1. Define $\frac{\|\mathbf{x}_i\|_1}{\|\mathbf{x}_i\|_2} \leq \varphi$ with $1 \leq \varphi \leq \sqrt{d}$, and $\|\mathbf{x}_i\|_2 \leq \tau$ for all $i \in [n]$. Then, with probability at least $1-\eta-\delta$ we have*

$$\begin{aligned}\|\mathbf{C}_e - \mathbf{C}\|_2 \leq & \min\{\widetilde{O}\Big(f + \frac{\tau^2\varphi}{m}\sqrt{\frac{1}{n}} + \tau^2\sqrt{\frac{1}{nm}}\Big), \\ & \widetilde{O}\Big(f + \frac{\tau\varphi}{m}\sqrt{\frac{d\|\mathbf{C}\|_2}{n}} + \tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}}\Big)\},\end{aligned} \tag{5.3}$$

where $f = \frac{\tau^2}{n} + \frac{\tau^2\varphi^2}{nm} + \tau\varphi\sqrt{\frac{\|\mathbf{C}\|_2}{nm}}$, and $\widetilde{O}(\cdot)$ hides the logarithmic factors on η, δ, m, n, d, and α.

The formulation above explores that $1 \leq \|\mathbf{x}_i\|_1/\|\mathbf{x}_i\|_2 \leq \sqrt{d}$ by the Cauchy-Schwarz inequality. Before proceeding, we make several remarks to make a comparison with the following representative work: *Gauss-Inverse*, *UniSample-HD*, and *Sparse*. The first two methods provide error analysis without assuming data distribution, which is shown in [10, 16] and illustrated in our *appendix*. In the following remarks, only our method is sensitive to φ, and we also employ the fact that $\frac{1}{nd}\|\mathbf{X}\|_F^2 \leq \|\mathbf{C}\|_2 \leq \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 = \tau^2$ to simplify all asymptotic bounds.

Remark 5.1. Eq. (5.3) with $\varphi = \sqrt{d}$ indicates the error bound for our estimator $\mathbf{C}_e$ in the worst case, where the magnitudes of each entry in all of the input data vectors are the same (i.e., highly uniformly distributed). Even in this case, our error bound has a leading term of order $\min\{\widetilde{O}\Big(\frac{\tau^2 d}{nm} + \tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2}{m}\sqrt{\frac{d}{n}}\Big), \widetilde{O}\Big(\frac{\tau^2 d}{nm} + \frac{\tau d}{m}\sqrt{\frac{\|\mathbf{C}\|_2}{n}}\Big)\}$, which is the same as *Gauss-Inverse* ignoring logarithmic factors. In contrast, as the magnitudes of the entries in each data vector become uneven, φ gets smaller, leading to a tighter error bound than that in *Gauss-Inverse*. Furthermore, when most of the entries in each vector $\mathbf{x}_i$ have very low magnitudes, the summation of these magnitudes will be comparable to a particular constant. This situation is typical because in practice only a limited number of features in each input data dominate the learning performance. Hence, φ turns to $O(1)$, and Eq. (5.3) becomes $\min\{\widetilde{O}\Big(\frac{\tau^2}{n} + \tau^2\sqrt{\frac{1}{nm}}\Big), \widetilde{O}\Big(\frac{\tau^2}{n} + \tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}}\Big)\}$, which is tighter than the leading term of *Gauss-Inverse* by a factor of at least $\sqrt{d/m}$. As explained in the next section, *Gauss-Inverse* also lacks computational efficiency.

Remark 5.2. As our target is to compress data to a smaller m that is not comparable to d in practice, $O(d-m)$ can be approximately regarded as $O(d)$. Then, the error of *UniSample-HD* is $\widetilde{O}\Big(\frac{\tau^2 d}{nm} + \tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2 d}{m}\sqrt{\frac{1}{nm}}\Big)$, which is asymptotically worse than our bound. When n is sufficiently large, the leading term of its error becomes $\widetilde{O}\Big(\tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2 d}{m}\sqrt{\frac{1}{nm}}\Big)$, which can be weaker than the leading term in our method by a factor of 1 to $\sqrt{d/m}$ when $\varphi = \sqrt{d}$, and at least d/m when $\varphi = O(1)$.

However, if m is sufficiently close to d, which is not meaningful for practical usage, $O(d-m) = O(1)$ will hold and the error of *UniSample-HD* becomes $\widetilde{O}\Big(\frac{\tau^2 d}{nm} + \tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2}{m}\sqrt{\frac{d}{nm}}\Big)$. This bound may slightly outperform ours by a factor of $\sqrt{d/m} = O(1)$ when $\varphi = \sqrt{d}$, but is still worse than ours when $\varphi = O(1)$. These results also coincide with the fact that *UniSample-HD* adopts uniform sampling *without replacement* combined with the Hadamard matrix, but we employ weighted sampling *with replacement*.

Remark 5.3. The *Sparse* method, which employs a sparse matrix for each $\mathbf{S}_i$, is not sufficiently accurate as demonstrated in our experiments. Moreover, there is no error analysis available for its unbiased estimator to characterize the estimation error versus the compression ratio.

Thus far, we have not made statistical nor structural assumptions concerning the input data or covariance matrix to derive our provable results. However, motivated by [16], it is also straightforward to extend our results to the statistical data and a low-rank covariance matrix estimation. The derived results below are polynomially equivalent to those in *Gauss-Inverse* [16]. Corollary 5.2 shows the (low-rank) covariance matrix estimation on Gaussian data, and Corollary 5.3 indicates the derived covariance estimator also guarantees the accuracy of the principal components regarding the subspace learning.

Corollary 5.2. *Given $\mathbf{X} \in \mathbb{R}^{d\times n}$ $(2 \leq d)$ and an unknown population covariance matrix $\mathbf{C}_p \in \mathbb{R}^{d\times d}$ with each column vector $\mathbf{x}_i \in \mathbb{R}^d$ i.i.d. generated from the Gaussian distribution*

$\mathcal{N}(\mathbf{0}, \mathbf{C}_p)$. *Let* $\mathbf{C}_e$ *be constructed by Algorithm 5.1 with the sampling size* $2 \leq m < d$. *Then, with probability at least* $1 - \eta - \delta - \zeta$,

$$\frac{\|\mathbf{C}_e - \mathbf{C}_p\|_2}{\|\mathbf{C}_p\|_2} \leq \widetilde{O}\Big(\frac{d^2}{nm} + \frac{d}{m}\sqrt{\frac{d}{n}}\Big); \tag{5.4}$$

Additionally, assuming rank$(\mathbf{C}_p) \leq r$, *with probability at least* $1 - \eta - \delta - \zeta$ *we have*

$$\frac{\|[\mathbf{C}_e]_r - \mathbf{C}_p\|_2}{\|\mathbf{C}_p\|_2} \leq \widetilde{O}\Big(\frac{rd}{nm} + \frac{r}{m}\sqrt{\frac{d}{n}} + \sqrt{\frac{rd}{nm}}\Big), \tag{5.5}$$

where $[\mathbf{C}_e]_r$ *is the solution to* $\min_{rank(A) \leq r} \|\mathbf{A} - \mathbf{C}_e\|_2$, *and* $\widetilde{O}(\cdot)$ *hides the logarithmic factors on* η, δ, ζ, m, n, d, *and* α.

Corollary 5.3. *Given* $\mathbf{X}$, d, m, $\mathbf{C}_p$ *and* $\mathbf{C}_e$ *as defined in Corollary 5.2. Let* $\prod_k = \sum_{i=1}^k \mathbf{u}_i \mathbf{u}_i^T$ *and* $\widehat{\prod}_k = \sum_{i=1}^k \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i^T$ *with* $\{\mathbf{u}_i\}_{i=1}^k$ *and* $\{\hat{\mathbf{u}}_i\}_{i=1}^k$ *being the leading* k *eigenvectors of* $\mathbf{C}_p$ *and* $\mathbf{C}_e$, *respectively. Denote by* λ_k *the* k*-th largest eigenvalue of* $\mathbf{C}_p$. *Then, with probability at least* $1 - \eta - \delta - \zeta$,

$$\frac{\|\widehat{\prod}_k - \prod_k\|_2}{\|\mathbf{C}_p\|_2} \leq \frac{1}{\lambda_k - \lambda_{k+1}} \widetilde{O}\Big(\frac{d^2}{nm} + \frac{d}{m}\sqrt{\frac{d}{n}}\Big), \tag{5.6}$$

where the eigengap $\lambda_k - \lambda_{k+1} > 0$ *and* $\widetilde{O}(\cdot)$ *hides the logarithmic factors on* η, δ, ζ, m, n, d, *and* α.

The proof details of all our theoretical results are relegated to the *appendix*. We leverage the Matrix Bernstein inequality [171] and establish the error bound of our proposed estimator on an arbitrary sampling probability in order to determine which sampling probability brings the best estimation accuracy. The employment of the Matrix Bernstein inequality involves controlling the range and variance of all zero-mean random matrices, whose derivations differ from those in [10, 16] because of different data compression schemes. Moreover, to obtain the desired tight bound for the range and variance, we precisely provide a group of closed-form equalities or concentration inequalities for various quantities (see our proposed **Lemma 1** and **Lemma 2** along with their proofs in Section 5.6).

5.3.3 Computational Complexity

Recall that we have n data samples in the d-dimensional space, and let m be the target compressed dimension. Regarding estimating $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$, the computational comparisons between our method and the representative baseline methods are presented in Table 5.1, in which *Standard* method means that we compute $\mathbf{C}$ directly without data compression. For the definition of covariance matrix $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T - \bar{\mathbf{x}}\bar{\mathbf{x}}^T$, extra computational costs (i.e., $O(gd)$ storage, $O(gd)$ communication cost, and $O(nd)$ time) must be added to the last four compression methods in the table, where $g \ll n$ is the number of the entire remote sites. All detailed analysis is relegated to the *appendix*.

Table 5.1: Computational costs on the storage, communication, and time.

Method	Storage	Comm.	Time
Standard	$O(nd + d^2)$	$O(nd)$	$O(nd^2)$
Gauss-Inverse	$O(nm + d^2)$	$O(nm)$	$O(nmd + nm^2d + nd^2) + T_G$
Sparse	$O(nm + d^2)$	$O(nm)$	$O(d + nm^2) + T_S$
UniSample-HD	$O(nm + d^2)$	$O(nm)$	$O(nd \log d + nm^2)$
Our method	$O(nm + d^2)$	$O(nm)$	$O(nd + nm \log d + nm^2)$

T_G and T_S in Table 5.1 represent the time taken to generate the Gaussian matrices and sparse matrices by fast pseudo-random number generators like Mersenne Twister [137], which can be enormous [10] and proportional to nmd and nd^2, respectively, up to certain small constants. Hence, our method can be regarded as the most efficient when d is large. Furthermore, by using the smallest m to obtain the same estimation accuracy as the other methods, our approach incurs the least computational burden.

5.4 Empirical Studies

In this section, we empirically verify the properties of the proposed method and demonstrate its superiority. We compare its estimation accuracy with that of *Gauss-Inverse*, *Sparse*, and *UniSample-HD*. We also report the time comparisons.

We run all algorithms on both synthetic and real-world datasets whose largest dimension is around and below 10^5. Such dimension is not very high in modern data analysis, but this limitation is due to that reporting the estimation error by calculating the spectral norm of a covariance matrix with its size larger than $10^5 \times 10^5$ will take intolerable amount of memory and time. The parameter selection on α is deferred to the *appendix*, and we empirically set $\alpha = 0.9$. To allow a fair comparison of the time consumptions measured by FLOPS, we implement all algorithms in C++ and run them in a single thread mode on a standard workstation with Intel CPU@2.90GHz and 128GB RAM.

5.4.1 Experiments on Synthetic Datasets

To clearly examine the performance, we compare all methods on six synthetic datasets: $\{\mathbf{X}_i\}_{i=1}^3 \in \mathbb{R}^{1024\times 20000}$, $\mathbf{X}_4 \in \mathbb{R}^{1024\times 200000}$, $\mathbf{X}_5 \in \mathbb{R}^{2048\times 200000}$, and $\mathbf{X}_6 \in \mathbb{R}^{65536\times 200000}$, which are generated based on the generative model [118]. Specifically, given a matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ from such model, it is formally defined as $\mathbf{X} = \mathbf{UFG}$, where $\mathbf{U} \in \mathbb{R}^{d\times k}$ defines the signal column space with $\mathbf{U}^T\mathbf{U} = \mathbf{I}_k$ ($k \leq d$), the square diagonal matrix $\mathbf{F} \in \mathbb{R}^{k\times k}$ contains the diagonal entries $f_{ii} = 1-(i-1)/k$ that gives linearly diminishing signal singular values, and $\mathbf{G} \in \mathbb{R}^{k\times n}$ is the signal coefficient with $g_{ij} \sim \mathcal{N}(0,1)$ that is the Gaussian distribution. We let $k \approx 0.005d$, then setting $d = 1024$ and $n = 20000$ completes the creation of data $\mathbf{X}_1$. For $\mathbf{X}_2$, it is defined as $\mathbf{DX}$, where each entry in the square diagonal matrix $\mathbf{D}$ is defined by $d_{ii} = 1/\beta_i$, and β_i is randomly sampled from the integer set [15]. Regarding $\mathbf{X}_3$, it is constructed in the same way as $\mathbf{X}_1$ except that $\mathbf{F}$ now becomes an identity matrix. Next, $\{\mathbf{X}_i\}_{i=4}^{6}$ follow the same generation strategy of $\mathbf{X}_2$ except for the n and d.

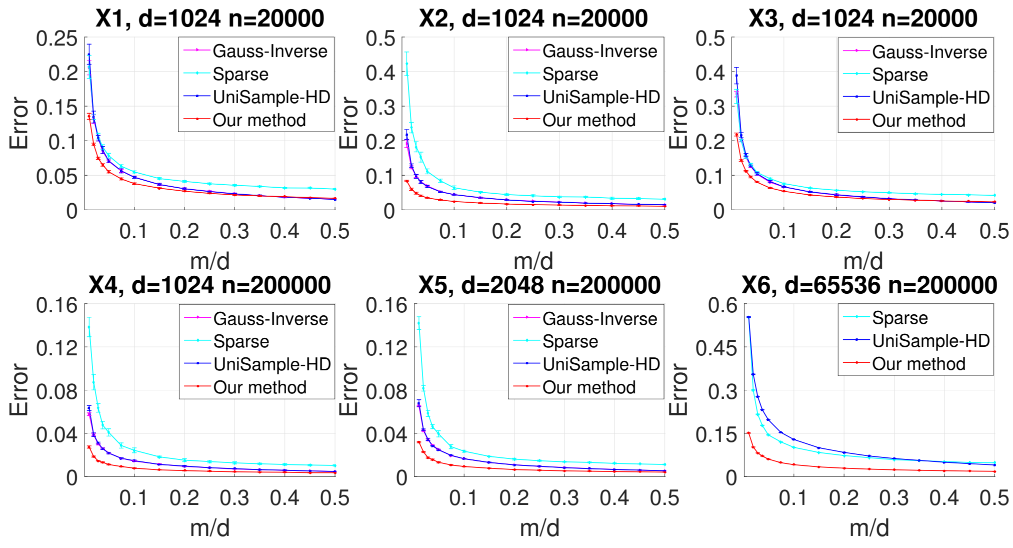

Figure 5.1: Accuracy comparisons of covariance matrix estimation on synthetic datasets. The estimation error is measured by $\|\mathbf{C}_e - \mathbf{C}\|_2/\|\mathbf{C}\|_2$ with $\mathbf{C}_e$ calculated by all compared methods, and cf $= m/d$ is the compression ratio.

In Figure 5.1, we plot the relative estimation error averaged over ten runs with its standard deviation versus the naive compression ratio cf $= m/d$. Note that a large cf is not necessary for practical usage, and our aim is to compress data to a smaller volume. In Figure 5.2, we report the running time taken in both the compressing and recovering stages, which preliminarily depicts the efficiency of the different methods and indicates how much power should be spent in the practical computation.

Generally, our method displays the least error and deviation for all datasets and its error decreases dramatically with an increase at a small cf. This observation indicates that our method can achieve sufficient estimation accuracy by using substantially fewer data entries than the other methods. For $\mathbf{X}_1$ ($\varphi = 0.81\sqrt{d}$), the magnitudes of the data entries are highly uniformly distributed, and thus our method can be regarded as uniform sampling with replacement, which may perform slightly worse than *UniSample-HD* and *Gauss-Inverse* if cf becomes large enough. After allowing the magnitudes to vary within a moderately larger range in $\mathbf{X}_2$ ($\varphi = 0.55\sqrt{d}$), our method considerably outperforms the other three methods. Its improvement comes from that *only* our method is sensitive to φ and a smaller φ produces a tighter result, as demonstrated by Remarks 5.1 and 5.2. However, the error of each method in $\mathbf{X}_3$ ($\tau/\sqrt{\|\mathbf{C}\|_2} = 5.5, \varphi = 0.81\sqrt{d}$) is larger than that in $\mathbf{X}_1$ ($\tau/\sqrt{\|\mathbf{C}\|_2} = 4.3, \varphi = 0.81\sqrt{d}$), respectively. It is because of that almost *all* methods are sensitive to $\tau/\sqrt{\|\mathbf{C}\|_2}$, and the error $\|\mathbf{C}_e - \mathbf{C}\|_2/\|\mathbf{C}\|_2$ increases when $\tau/\sqrt{\|\mathbf{C}\|_2}$ rises. Such phenomenon is demonstrated via dividing numerous error bounds in Remarks 5.1 and 5.2 by $\|\mathbf{C}\|_2$. Our method also achieves the best performance in $\mathbf{X}_4$. Although the φ and $\tau/\sqrt{\|\mathbf{C}\|_2}$ in $\mathbf{X}_4$ are approximately equal with those in $\mathbf{X}_2$, yet the proved error bounds with Remarks 5.1 and 5.2 reveal that a larger n in $\mathbf{X}_4$ will lead to smaller estimation errors given the same cf. Finally, our method also achieves the best accuracy when the dimension d increases in both $\mathbf{X}_5$ and $\mathbf{X}_6$. Besides, taking more data (i.e., enlarging n) as suggested by $\mathbf{X}_4$ can be considered to reduce the error in $\mathbf{X}_5$ and $\mathbf{X}_6$. Note that *Gauss-Inverse* has not been run on $\mathbf{X}_6$ since it costs enormous time.

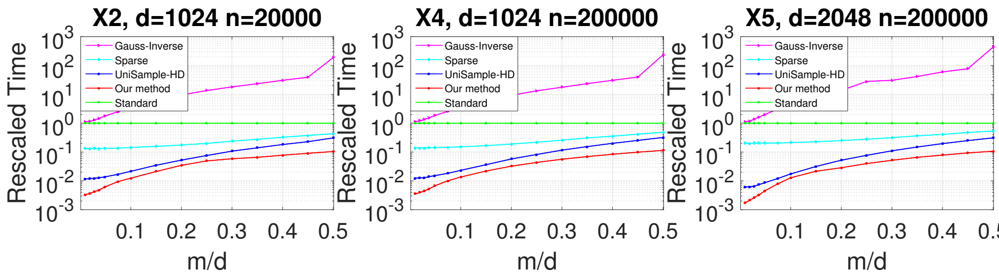

Figure 5.2: Time comparisons of covariance matrix estimation on synthetic datasets. Rescaled time results from the running time normalized by that spent in the *Standard* way of calculating $\mathbf{C} = \mathbf{X}\mathbf{X}^T/n$ without data compression, and it is plotted in log scale.

Turning to *Gauss-Inverse*, it becomes highly accurate when cf increases but requires much more time than *Standard* (see Figure 5.2) so that its usage might be ruled out in practice. However, *Gauss-Inverse* remains a good choice when we are in urgent need of reducing the storage and communication burden. *Sparse*, which has no error analysis of its unbiased estimator, generally performs less accurately than the others but requires less time than *Standard*. *UniSample-HD* is efficient while it still consumes more time than our method. Also, its accuracy is inferior to our method especially when cf is small. In conclusion, our method is capable of compressing data to a very small size while guaranteeing both estimation accuracy and computational efficiency.

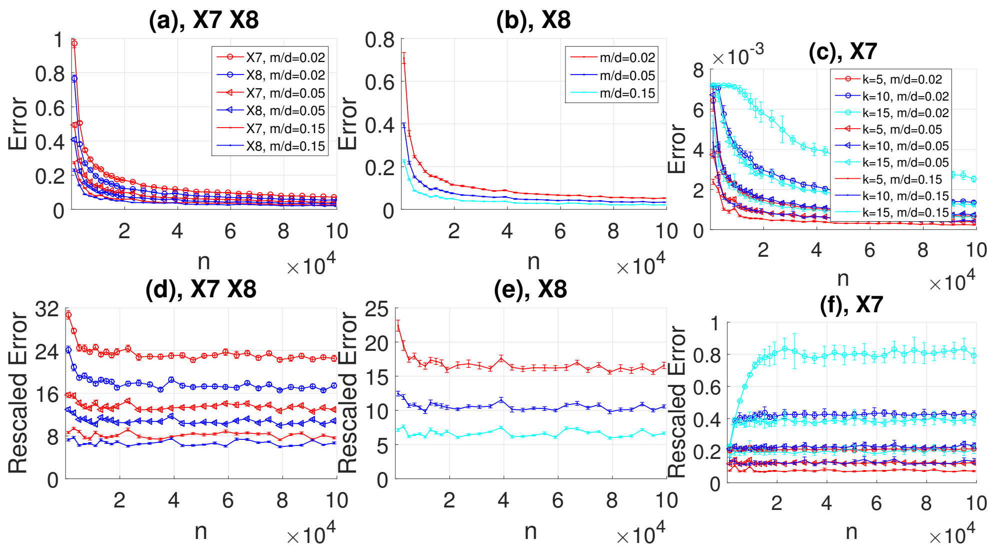

Figure 5.3: Convergence rates of our method for the settings in Corollaries 5.2 and 5.3.

As confirmed in Figure 5.1, a large n benefits the estimation accuracy. Thus, we study its effect more quantitatively. We conduct experiments following the settings as defined in Corollaries 5.2 and 5.3, and their results in Eqs. (5.4)-(5.6) clearly show that the errors decay in $1/\sqrt{n}$ convergence rate if $d \ll n$. We run our method on another two synthetic datasets $\{\mathbf{X}_t\}_{t=7}^{8} \in \mathbb{R}^{d\times n}$ that follow the d-dimensional multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{C}_{pt})$, where the (i, j)-th element of $\mathbf{C}_{p7} \in \mathbb{R}^{d\times d}$ is $0.5^{|i-j|/50}$, and $\mathbf{C}_{p8} \in \mathbb{R}^{d\times d}$ is a low-rank matrix that satisfies $\min_{\text{rank}(\mathbf{A})\le r} \|\mathbf{A} - \mathbf{C}_{p7}\|_2$. We take $d = 1000$, $r = 5$, $m/d = \{0.02, 0.05, 0.15\}$, $k = \{5, 10, 15\}$, and vary n from 1000 to 100000. In Figure 5.3, the top three plots report the errors as defined in the LHS of Eqs. (5.4)-(5.6), respectively. Then, dividing such errors by $1/\sqrt{n}$ obtains the bottom three plots accordingly.

The observation that the curves in plots (d)-(f) are roughly flat validates that the error bounds induced by our method decay rapidly with n in the $1/\sqrt{n}$ convergence rate, which coincides with Eqs. (5.4)-(5.6). In addition to the fast error convergence for the low-rank matrix $\mathbf{C}_{p8}$, our method can also obtain an increasingly better estimation accuracy for a high-rank covariance matrix $\mathbf{C}_{p7}$ if we enlarge n, which is displayed in plot (a). Besides, considering the omitted plot where the eigengap $\lambda_k - \lambda_{k+1}$ of $\mathbf{C}_{p7}$ decreases with k, the fact that the errors in plot (c) increase with k also coheres with Eq. (5.6). To conclude, our method also adapts well to the specific settings in Corollaries 5.2 and 5.3, and all induced error bounds indeed satisfy a $1/\sqrt{n}$ convergence rate.

5.4.2 Experiments on Real-World Datasets

In the second set of experiments, we use nine publicly available real-world datasets [33, 23, 8], some of which are gathered from many distributed sensors. Their statistics are displayed in Figure 5.4. We again compare the estimation accuracy of the proposed method against the other three approaches. As can be seen from the figure, our method consistently exhibits superior accuracy over all cf $= m/d$, and its error decreases dramatically when cf grows. The error of the other three methods also decreases with cf but is still large at a small cf. Besides, our method enjoys the least deviation. In summary, these results confirm that our method can compress data to the lowest volume with the best accuracy, thereby substantially reducing storage, communication, and processing time cost in practice.

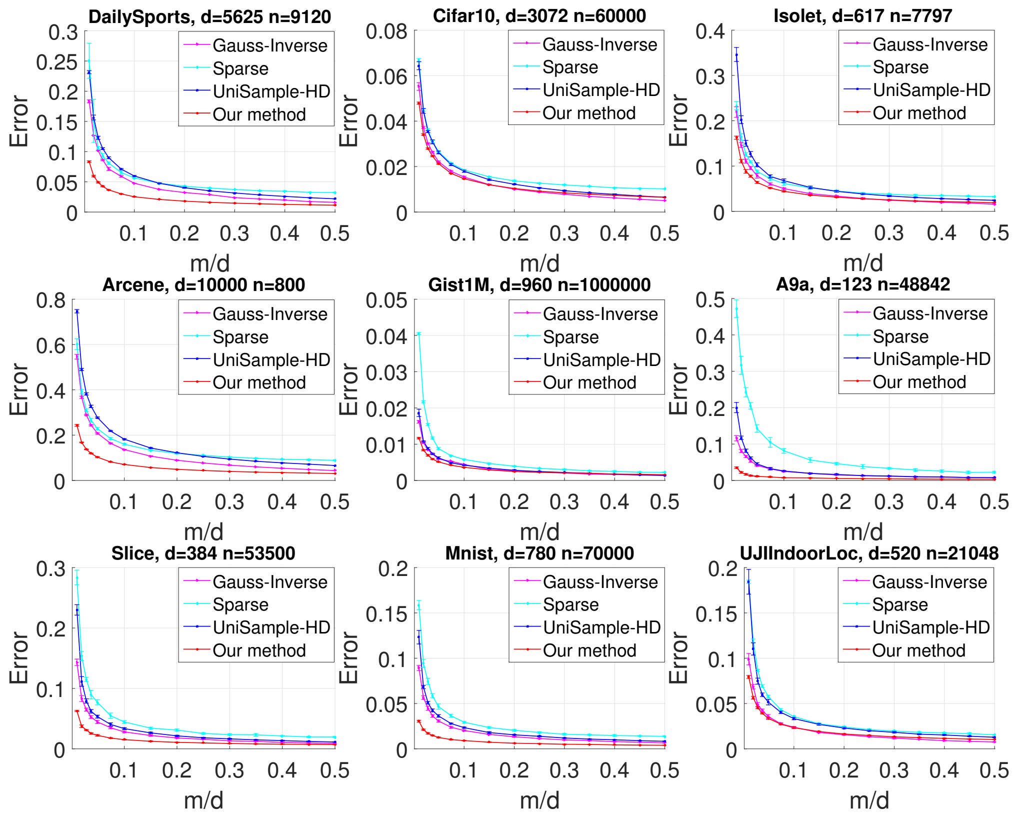

Figure 5.4: Accuracy comparisons of covariance matrix estimation on real-world datasets.

5.5 Conclusion

In this chapter, we describe a weighted sampling method for accurate and efficient calculation of an unbiased covariance matrix estimator. The analysis demonstrates that our method can employ a smaller data volume than the other approaches to achieve an equal accuracy, and is highly efficient regarding the communication, storage, and processing time. The empirical results of the algorithm's application to both synthetic and real-world datasets further support our analysis and demonstrate its significant improvements over other state-of-the-art methods.

Compared with the sampling-with-replacement scheme in this chapter, we seek to make more achievements via a sampling-without-replacement scheme in the future work. Analyzing the corresponding unbiased estimator will pose significant technical challenges in this research direction.

5.6 Proofs

5.6.1 Preliminaries

First, we first show the Matrix Bernstein inequality employed for characterizing the sums of independent random variables/matrices, and then present a matrix perturbation result for eigenvalues.

Theorem 5.3 ([171, p. 76])**.** *Let* $\{\mathbf{A}_i\}_{i=1}^{L} \in \mathbb{R}^{d\times n}$ *be independent random matrices with* $\mathbb{E}\left[\mathbf{A}_i\right] = 0$ *and* $\|\mathbf{A}_i\|_2 \leq R$*. Define* $\sigma^2 = \max\{\|\sum_{i=1}^{L}\mathbb{E}\left[\mathbf{A}_i\mathbf{A}_i^T\right]\|_2, \|\sum_{i=1}^{L}\mathbb{E}\left[\mathbf{A}_i^T\mathbf{A}_i\right]\|_2\}$ *as the variance. Then for all* $\epsilon \geq 0$*,*

$$\mathbb{P}(\|\sum_{i=1}^{L}\mathbf{A}_i\|_2 \geq \epsilon) \leq (d+n)\exp(\frac{-\epsilon^2/2}{\sigma^2 + R\epsilon/3}). \tag{5.7}$$

Theorem 5.4 ([75, p. 396])**.** *If* $\mathbf{A} \in \mathbb{R}^{d\times d}$ *and* $\mathbf{A}+\mathbf{E} \in \mathbb{R}^{d\times d}$ *are symmetric matrices, then*

$$\lambda_k(\mathbf{A}) + \lambda_d(\mathbf{E}) \leq \lambda_k(\mathbf{A}+\mathbf{E}) \leq \lambda_k(\mathbf{A}) + \lambda_1(\mathbf{E}) \tag{5.8}$$

for $k \in [d]$*, where* $\lambda_k(\mathbf{A}+\mathbf{E})$ *and* $\lambda_k(\mathbf{A})$ *designate the* k*-th largest eigenvalues.*

Next, we propose two lemmas: Lemma 5.5 and Lemma 5.6, which are used to prove the foregoing theorems. The details are described below.

Lemma 5.5. *Given any vector* $\mathbf{x} \in \mathbb{R}^d$*, and* $m < d$*, sample* m *entries from* $\mathbf{x}$ *with replacement by running Algorithm 5.1 with the inputs* $\mathbf{x}$ *and* m*. Let* $\{p_k\}_{k=1}^d$ *denote the corresponding sampling probabilities,* $\mathbf{S} \in \mathbb{R}^{d\times m}$ *denote the corresponding rescaled sampling matrix, and* $\{\mathbf{e}_k\}_{k=1}^d$ *denote the standard basis vectors for* $\mathbb{R}^d$*. Then, we have*

$$\mathbb{E}\left[\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T\right] = \sum_{k=1}^d \frac{x_k^2}{mp_k}\mathbf{e}_k\mathbf{e}_k^T + \frac{m-1}{m}\mathbf{x}\mathbf{x}^T; \tag{5.9}$$

$$\mathbb{E}\left[\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T)\right] = \sum_{k=1}^d (\frac{1}{mp_k} + \frac{m-1}{m})x_k^2\mathbf{e}_k\mathbf{e}_k^T; \tag{5.10}$$

$$\begin{aligned}&\mathbb{E}\left[(\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T))^2\right]\\ &= \sum_{k=1}^d \left[\frac{1}{m^3p_k^3} + \frac{7(m-1)}{m^3p_k^2} + \frac{6(m^2-3m+2)}{m^3p_k}\right.\\ &\quad \left. + \frac{m^3-6m^2+11m-6}{m^3}\right] x_k^4\mathbf{e}_k\mathbf{e}_k^T;\end{aligned} \tag{5.11}$$

$$\begin{aligned}&\mathbb{E}\left[\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T)\right]\\ &= (\mathbb{E}\left[\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T)\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T\right])^T\\ &= \sum_{k=1}^d \left[\frac{1}{m^3p_k^3} + \frac{6(m-1)}{m^3p_k^2} + \frac{3(m^2-3m+2)}{m^3p_k}\right] x_k^4\mathbf{e}_k\mathbf{e}_k^T\\ &\quad + \frac{m-1}{m^3}\mathbf{x}\mathbf{x}^T\mathbb{D}(\{\frac{x_k^2}{p_k^2}\})\\ &\quad + \frac{3(m^2-3m+2)}{m^3}\mathbf{x}\mathbf{x}^T\left[\mathbb{D}(\{\frac{x_k^2}{p_k}\}) + \frac{m-3}{3}\mathbb{D}(\{x_k^2\})\right];\end{aligned} \tag{5.12}$$

$$\begin{aligned}&\mathbb{E}\left[(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T)^2\right]\\ &= \sum_{k=1}^d \left[\frac{4(m-1)}{m^3p_k^2} + \frac{1}{m^3p_k^3}\right] x_k^4\mathbf{e}_k\mathbf{e}_k^T\end{aligned}$$

$$+\sum_{k=1}^{d}\left[\frac{\|\mathbf{x}\|_2^2(m^2-3m+2)}{m^3}+\frac{m-1}{m^3}\sum_{k=1}^{d}\frac{x_k^2}{p_k}\right]\frac{x_k^2}{p_k}\mathbf{e}_k\mathbf{e}_k^T$$
$$+\left[\frac{\|\mathbf{x}\|_2^2(m^3-6m^2+11m-6)}{m^3}+\frac{m^2-3m+2}{m^3}\sum_{k=1}^{d}\frac{x_k^2}{p_k}\right]\mathbf{x}\mathbf{x}^T$$
$$+\mathbf{x}\mathbf{x}^T\left[\frac{2(m^2-3m+2)}{m^3}\mathbb{D}(\{\frac{x_k^2}{p_k}\})+\frac{m-1}{m^3}\mathbb{D}(\{\frac{x_k^2}{p_k^2}\})\right]$$
$$+\left[\frac{2(m^2-3m+2)}{m^3}\mathbb{D}(\{\frac{x_k^2}{p_k}\})+\frac{m-1}{m^3}\mathbb{D}(\{\frac{x_k^2}{p_k^2}\})\right]\mathbf{x}\mathbf{x}^T, \quad (5.13)$$

where the expectation is w.r.t. $\mathbf{S}$*, and* $\mathbb{D}(\{x_k^2\})$ *denotes a square diagonal matrix with* $\{x_k^2\}_{k=1}^d$ *on its diagonal that can be extended to other similar notations.*

Lemma 5.6. *Given the definitions in Lemma 5.5. Then, with probability at least* $1-\sum_{k=1}^d \eta_k$*, we have*

$$\|\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}\mathbf{S}\mathbf{S}^T\|_2 \leq \sum_{k\in\Gamma} f^2(x_k,\eta_k,m), \quad (5.14)$$

where Γ *is a set containing at most* m *different elements of* $[d]$ *with its cardinality* $|\Gamma| \leq m$*, and* $f(x_k,\eta_k,m) = |x_k| + \log(\frac{2}{\eta_k})\left[\frac{|x_k|}{3mp_k}+|x_k|\sqrt{\frac{1}{9m^2p_k^2}+\frac{2}{\log(2/\eta_k)}(\frac{1}{mp_k}-\frac{1}{m})}\right]$.

Remark 5.4. For the expressions in Lemma 5.5 and Lemma 5.6, the sampling probability p_k appears in the denominator, which indicates that the derived bound may be sensitive to a highly small $p_k \neq 0$. However, in terms of any $p_k = 0$, we can define $\frac{|x_k|^a}{p_k^b} = 0$ for $a, b > 0$, because we follow the rule that $p_k = 0$ only when $x_k = 0$ and $x_k = 0$ can never be sampled. Thus, the aforementioned two lemmas and other derived results are applicable to the case where there exists $p_k = 0$.

5.6.2 Proof of Lemma 5.5

Proof. According to Algorithm 5.1, each column vector in the rescaled sampling matrix $\mathbf{S} \in \mathbb{R}^{d\times m}$ is sampled with replacement from $\{\mathbf{r}_k = \frac{1}{\sqrt{mp_k}}\mathbf{e}_k\}_{k=1}^d$ with corresponding probabilities $\{p_k\}_{k=1}^d$, where $\{\mathbf{e}_k\}_{k=1}^d$ are the standard basis vectors for $\mathbb{R}^d$.

Firstly, we prove Eq. (5.9). By the definition, we expand

$$\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T = \sum_{j=1}^m \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{x}\sum_{j=1}^m \mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \tag{5.15}$$

$$= \sum_{j=1}^m \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T + \sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T, \tag{5.16}$$

where the random variable t_j is in $[d]$.

Passing the expectation over $\mathbf{S}$ through the sum in Eq. (5.16), we have

$$\begin{aligned}\mathbb{E}\sum_{j=1}^m \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T &= \sum_{j=1}^m\sum_{k=1}^d \mathbb{P}(t_j=k)\mathbf{r}_k\mathbf{r}_k^T\mathbf{x}\mathbf{x}^T\mathbf{r}_k\mathbf{r}_k^T \\ &= \sum_{j=1}^m\sum_{k=1}^d p_k\frac{1}{m^2p_k^2}\mathbf{e}_k\mathbf{e}_k^T\mathbf{x}\mathbf{x}^T\mathbf{e}_k\mathbf{e}_k^T \\ &= \sum_{k=1}^d \frac{x_k^2}{mp_k}\mathbf{e}_k\mathbf{e}_k^T, \end{aligned}\tag{5.17}$$

and similarly

$$\mathbb{E}\sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \tag{5.18}$$

$$= \sum_{i\neq j\in[m]}\sum_{k=1}^d\sum_{q=1}^d \mathbb{P}(t_i=k)\mathbb{P}(t_j=q)\mathbf{r}_k\mathbf{r}_k^T\mathbf{x}\mathbf{x}^T\mathbf{r}_q\mathbf{r}_q^T \tag{5.19}$$

$$= \sum_{k=1}^d\sum_{q=1}^d x_kx_q\frac{m-1}{m}\mathbf{e}_k\mathbf{e}_q^T \tag{5.20}$$

$$= \frac{m-1}{m}\mathbf{x}\mathbf{x}^T. \tag{5.21}$$

Now, combing Eq. (5.17) with Eq. (5.21) immediately proves Eq. (5.9).

Then, Eq. (5.10) can be proved based on Eq. (5.9) by

$$\begin{aligned}\mathbb{E}\left[\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T)\right] &= \mathbb{D}(\mathbb{E}\left[\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T\right]) &(5.22)\\ &= \sum_{k=1}^{d}(\frac{1}{mp_k}+\frac{m-1}{m})x_k^2\mathbf{e}_k\mathbf{e}_k^T. &(5.23)\end{aligned}$$

Alternatively, $\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T)$ can be explicitly expanded by

$$\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T) = \sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T. \tag{5.24}$$

Thus, the whole target expectations in Eq. (5.11), Eq. (5.12) and Eq. (5.13) can be explicitly expanded, and we can use similar ways of proving Eq. (5.9) to prove the remainder of the lemma.

To prove Eq. (5.11), we expand

$$\begin{aligned}&\mathbb{E}\left[(\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T))^2\right]\\ &= \mathbb{E}\left[(\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T)^2\right] &(5.25)\\ &= \mathbb{E}\left[\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\right] &(5.26)\\ &= \mathbb{E}\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T &(5.27)\end{aligned}$$

$$+\mathbb{E}\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \tag{5.28}$$

$$+\mathbb{E}\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \tag{5.29}$$

$$+\mathbb{E}\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T, \tag{5.30}$$

where the four terms in the last equations are calculated as:

$$\begin{aligned}
&Eq.\ (5.27)\\
&=\mathbb{E}\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\\
&=\mathbb{E}\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\\
&+\mathbb{E}\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\\
&=\sum_{k_1=1}^{d}\sum_{j=1}^{m}p_{k_1}\frac{1}{m^4p_{k_1}^4}\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\\
&\quad+\mathbb{E}\sum_{i\neq j\in[m]}\sum_{k_1=1}^{d}\sum_{q=1}^{d}p_{k_1}p_q\frac{1}{m^4p_{k_1}^2p_q^2}\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{e}_q\mathbf{e}_q^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_q\mathbf{e}_q^T\\
&=\sum_{k=1}^{d}\frac{x_k^4}{m^3p_k^3}\mathbf{e}_k\mathbf{e}_k^T+\sum_{k=1}^{d}\frac{(m^2-m)x_k^4}{m^4p_k^2}\mathbf{e}_k\mathbf{e}_k^T
\end{aligned}$$

$$= \sum_{k=1}^{d} (\frac{1}{m^3 p_k^3} + \frac{m-1}{m^3 p_k^2}) x_k^4 \mathbf{e}_k \mathbf{e}_k^T; \tag{5.31}$$

$Eq.$ (5.28)

$$= \mathbb{E} \sum_{j=1}^{m} \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \sum_{i \neq j \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T$$

$$= \mathbb{E} \sum_{g \neq i \neq j \in [m]} \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T$$

$$+ \mathbb{E} \sum_{g = i \neq j \in [m]} \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T$$

$$+ \mathbb{E} \sum_{g = j \neq i \in [m]} \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T$$

$$= \sum_{k_1, k_2, k_3 = 1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \mathbf{e}_{k_2} \mathbf{e}_{k_2}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{e}_{k_3} \mathbf{e}_{k_3}^T$$

$$+ \sum_{k_1, k_3 = 1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{e}_{k_3} \mathbf{e}_{k_3}^T$$

$$+ \sum_{k_1, k_2 = 1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \mathbf{e}_{k_2} \mathbf{e}_{k_2}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T$$

$$= \sum_{k_1=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} x_{k_1}^4 \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T$$

$$+ \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^4 \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T$$

$$+ \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^4 \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T$$

$$= \sum_{k=1}^{d} \left[\frac{m(m-1)(m-2)}{m^4 p_k} x_k^4 + \frac{2m(m-1)x_k^4}{m^4 p_k^2} \right] \mathbf{e}_k \mathbf{e}_k^T; \tag{5.32}$$

Eq. (5.29)

$$= \mathbb{E} \sum_{i \neq j \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \sum_{j=1}^{m} \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T$$

$$= \sum_{k=1}^{d} \left[\frac{m(m-1)(m-2)}{m^4 p_k} x_k^4 + \frac{2m(m-1)x_k^4}{m^4 p_k^2} \right] \mathbf{e}_k \mathbf{e}_k^T; \tag{5.33}$$

Eq. (5.30)

$$= \mathbb{E} \sum_{i \neq j \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \sum_{i \neq j \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T$$

$$= \mathbb{E} \sum_{i \neq j \neq g \neq h \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_h} \mathbf{s}_{t_h}^T$$

$$+ \mathbb{E} \sum_{i \neq j, i = g, j \neq h, g \neq h \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_h} \mathbf{s}_{t_h}^T$$

$$+ \mathbb{E} \sum_{i \neq j, i = h, j \neq g, g \neq h \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_h} \mathbf{s}_{t_h}^T$$

$$+ \mathbb{E} \sum_{i \neq j, i \neq g, j = h, g \neq h \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_h} \mathbf{s}_{t_h}^T$$

$$+ \mathbb{E} \sum_{i \neq j, i \neq h, j = g, g \neq h \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_h} \mathbf{s}_{t_h}^T$$

$$+ \mathbb{E} \sum_{i \neq j, i = g, j = h, g \neq h \in [m]} \mathbf{s}_{t_i} \mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \mathbf{s}_{t_g} \mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_h} \mathbf{s}_{t_h}^T$$

$$
\begin{aligned}
&+ \mathbb{E} \sum_{i \neq j, i=h, j=g, g \neq h \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k \mathbf{e}_k^T \mathbf{s}_{t_h}\mathbf{s}_{t_h}^T \\
&= \sum_{k=1}^{d} \left[\frac{m(m-1)(m-2)(m-3)}{m^4} x_k^4 + \frac{4m(m-1)(m-2)}{m^4 p_k} x_k^4 \right. \\
&\quad \left. + \frac{2m(m-1)}{m^4 p_k^2} x_k^4 \right] \mathbf{e}_k \mathbf{e}_k^T. \qquad (5.34)
\end{aligned}
$$

Combing the above terms with simplification and reformulation completes the proof of Eq. (5.11).

Now, we continue to prove Eq. (5.12).

$$
\begin{aligned}
&\mathbb{E}\left[\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T\mathbb{D}(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T)\right] \\
&= \mathbb{E}\left[\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x} \sum_{j=1}^{m} \mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\right] \\
&= \mathbb{E}\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \qquad (5.35) \\
&\quad + \mathbb{E}\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{i \neq j \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \qquad (5.36) \\
&\quad + \mathbb{E}\sum_{i \neq j \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \qquad (5.37) \\
&\quad + \mathbb{E}\sum_{i \neq j \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{i \neq j \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T, \qquad (5.38)
\end{aligned}
$$

where we calculate the four terms in the last equation as shown in below:

Eq. (5.35)

$$
= \mathbb{E}\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T
$$

$$= \mathbb{E}\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$+ \mathbb{E}\sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$= \sum_{k_1=1}^{d}\sum_{j=1}^{m} p_{k_1}\frac{1}{m^4p_{k_1}^4}\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{x}\mathbf{x}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T$$

$$+ \mathbb{E}\sum_{i\neq j\in[m]}\sum_{k_1=1}^{d}\sum_{q=1}^{d} p_{k_1}p_q\frac{1}{m^4p_{k_1}^2p_q^2}\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{x}\mathbf{x}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{e}_q\mathbf{e}_q^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_q\mathbf{e}_q^T$$

$$= \sum_{k=1}^{d}\frac{x_k^4}{m^3p_k^3}\mathbf{e}_k\mathbf{e}_k^T + \sum_{k=1}^{d}\frac{(m^2-m)x_k^4}{m^4p_k^2}\mathbf{e}_k\mathbf{e}_k^T$$

$$= \sum_{k=1}^{d}\left(\frac{1}{m^3p_k^3} + \frac{m-1}{m^3p_k^2}\right)x_k^4\mathbf{e}_k\mathbf{e}_k^T; \tag{5.39}$$

Eq. (5.36)

$$= \mathbb{E}\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$= \mathbb{E}\sum_{g\neq i\neq j\in[m]} \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$+ \mathbb{E}\sum_{g=i\neq j\in[m]} \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$+ \mathbb{E}\sum_{g=j\neq i\in[m]} \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$= \sum_{k_1,k_2,k_3=1}^{d}\frac{m(m-1)(m-2)}{m^4p_{k_1}}\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{x}\mathbf{x}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{e}_{k_2}\mathbf{e}_{k_2}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T$$

$$+ \sum_{k_1,k_3=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \mathbf{x}\mathbf{x}^T \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{e}_{k_3}\mathbf{e}_{k_3}^T$$

$$+ \sum_{k_1,k_2=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \mathbf{x}\mathbf{x}^T \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \mathbf{e}_{k_2}\mathbf{e}_{k_2}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T$$

$$= \sum_{k_1=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} x_{k_1}^4 \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T + \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^4 \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T$$

$$+ \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^4 \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T$$

$$= \sum_{k=1}^{d} \left[\frac{m(m-1)(m-2)}{m^4 p_k} x_k^4 + \frac{2m(m-1)}{m^4 p_k^2} x_k^4 \right] \mathbf{e}_k\mathbf{e}_k^T; \qquad (5.40)$$

Eq. (5.37)

$$= \mathbb{E} \sum_{i \neq j \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$= \mathbb{E} \sum_{i \neq j \neq g \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T$$

$$+ \mathbb{E} \sum_{i \neq j = g \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T$$

$$+ \mathbb{E} \sum_{i = g \neq j \in [m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{s}_{t_g}\mathbf{s}_{t_g}^T$$

$$= \sum_{k_1,k_2,k_3=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_3}} \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \mathbf{x}\mathbf{x}^T \mathbf{e}_{k_2}\mathbf{e}_{k_2}^T \mathbf{e}_{k_3}\mathbf{e}_{k_3}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{e}_{k_3}\mathbf{e}_{k_3}^T$$

$$+ \sum_{k_1,k_3=1}^{d} \frac{m(m-1)}{m^4 p_{k_3}^2} \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \mathbf{x}\mathbf{x}^T \mathbf{e}_{k_3}\mathbf{e}_{k_3}^T \mathbf{e}_{k_3}\mathbf{e}_{k_3}^T \sum_{k=1}^{d} x_k^2 \mathbf{e}_k\mathbf{e}_k^T \mathbf{e}_{k_3}\mathbf{e}_{k_3}^T$$

$$
\begin{aligned}
&+\sum_{k_2,k_3=1}^{d}\frac{m(m-1)}{m^4p_{k_3}^2}\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T\mathbf{x}\mathbf{x}^T\mathbf{e}_{k_2}\mathbf{e}_{k_2}^T\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T\\
=&\sum_{k_1,k_3=1}^{d}\frac{m(m-1)(m-2)}{m^4p_{k_3}}x_{k_1}x_{k_3}^3\mathbf{e}_{k_1}\mathbf{e}_{k_3}^T\\
&+\sum_{k_1,k_3=1}^{d}\frac{m(m-1)}{m^4p_{k_3}^2}x_{k_1}x_{k_3}^3\mathbf{e}_{k_1}\mathbf{e}_{k_3}^T+\sum_{k_3=1}^{d}\frac{m(m-1)}{m^4p_{k_3}^2}x_{k_3}^4\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T\\
=&\frac{m(m-1)(m-2)}{m^4}\mathbf{x}\mathbf{x}^T\mathbb{D}(\{\frac{x_k^2}{p_k}\})+\frac{m(m-1)}{m^4}\mathbf{x}\mathbf{x}^T\mathbb{D}(\{\frac{x_k^2}{p_k^2}\})\\
&+\frac{m(m-1)}{m^4}\sum_{k=1}^{d}\frac{x_k^4}{p_k^2}\mathbf{e}_k\mathbf{e}_k^T; \qquad (5.41)
\end{aligned}
$$

Eq. (5.38)

$$
\begin{aligned}
=&\,\mathbb{E}\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\\
=&\,\mathbb{E}\sum_{i\neq j\neq g\neq h\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T\\
&+\mathbb{E}\sum_{i\neq j,i=g,j\neq h,g\neq h\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T\\
&+\mathbb{E}\sum_{i\neq j,i=h,j\neq g,g\neq h\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T\\
&+\mathbb{E}\sum_{i\neq j,i\neq g,j=h,g\neq h\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T\\
&+\mathbb{E}\sum_{i\neq j,i\neq h,j=g,g\neq h\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T
\end{aligned}
$$

$$
\begin{aligned}
&+\mathbb{E}\sum_{i\neq j, i=g, j=h, g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T \\
&+\mathbb{E}\sum_{i\neq j, i=h, j=g, g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\sum_{k=1}^{d} x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T \\
=&\sum_{k_1,k_2,k_3,k_4=1}^{d}\frac{m(m-1)(m-2)(m-3)}{m^4}x_{k_1}x_{k_2}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_4}\mathbf{e}_{k_4}^T \\
&+\sum_{k_1,k_2,k_4=1}^{d}\frac{m(m-1)(m-2)}{m^4p_{k_1}}x_{k_1}x_{k_2}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_4}\mathbf{e}_{k_4}^T \\
&+\sum_{k_1,k_2,k_3=1}^{d}\frac{m(m-1)(m-2)}{m^4p_{k_1}}x_{k_1}x_{k_2}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \\
&+\sum_{k_1,k_2,k_3=1}^{d}\frac{m(m-1)(m-2)}{m^4p_{k_2}}x_{k_1}x_{k_2}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_2}\mathbf{e}_{k_2}^T \\
&+\sum_{k_1,k_2,k_4=1}^{d}\frac{m(m-1)(m-2)}{m^4p_{k_2}}x_{k_1}x_{k_2}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_2}\mathbf{e}_{k_2}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_4}\mathbf{e}_{k_4}^T \\
&+\sum_{k_1,k_2}^{d}\frac{m(m-1)}{m^4p_{k_1}p_{k_2}}x_{k_1}x_{k_2}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_2}\mathbf{e}_{k_2}^T \\
&+\sum_{k_1,k_2=1}^{d}\frac{m(m-1)}{m^4p_{k_1}p_{k_2}}x_{k_1}x_{k_2}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_2}\mathbf{e}_{k_2}^T\sum_{k=1}^{d}x_k^2\mathbf{e}_k\mathbf{e}_k^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \\
=&\sum_{k_1,k_2=1}^{d}\frac{m(m-1)(m-2)(m-3)}{m^4}x_{k_1}x_{k_2}^3\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T \\
&+\sum_{k_1=1}^{d}\frac{m(m-1)(m-2)}{m^4p_{k_1}}x_{k_1}^4\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T
\end{aligned}
$$

$$
\begin{aligned}
&+ \sum_{k_1=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} x_{k_1}^4 \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \\
&+ \sum_{k_1,k_2=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_2}} x_{k_1} x_{k_2}^3 \mathbf{e}_{k_1}\mathbf{e}_{k_2}^T \\
&+ \sum_{k_1,k_2=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_2}} x_{k_1} x_{k_2}^3 \mathbf{e}_{k_1}\mathbf{e}_{k_2}^T \\
&+ \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^4 \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \\
&+ \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^4 \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \\
= &\frac{m(m-1)(m-2)(m-3)}{m^4} \mathbf{x}\mathbf{x}^T \mathbb{D}(\{x_k^2\}) \\
&+ \frac{m(m-1)(m-2)}{m^4} \sum_{k=1}^{d} \frac{x_k^4}{p_k} \mathbf{e}_k\mathbf{e}_k^T \\
&+ \frac{m(m-1)(m-2)}{m^4} \sum_{k=1}^{d} \frac{x_k^4}{p_k} \mathbf{e}_k\mathbf{e}_k^T \\
&+ \frac{m(m-1)(m-2)}{m^4} \mathbf{x}\mathbf{x}^T \mathbb{D}(\{\frac{x_k^2}{p_k}\}) \\
&+ \frac{m(m-1)(m-2)}{m^4} \mathbf{x}\mathbf{x}^T \mathbb{D}(\{\frac{x_k^2}{p_k}\}) \\
&+ \frac{2m(m-1)}{m^4} \sum_{k=1}^{d} \frac{x_k^4}{p_k^2} \mathbf{e}_k\mathbf{e}_k^T. \qquad (5.42)
\end{aligned}
$$

Combing the above terms with simplification and reformulation completes the proof of Eq. (5.12).

Finally, we have to prove Eq. (5.13).

$$\mathbb{E}\left[(\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T)^2\right]$$

$$= \mathbb{E}\left[(\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x} \sum_{j=1}^{m} \mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T)^2 \right]$$

$$= \mathbb{E}(\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T + \sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T)^2$$

$$= \mathbb{E}(\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T)^2 \tag{5.43}$$

$$+ \mathbb{E}(\sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T)^2 \tag{5.44}$$

$$+ \mathbb{E}\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \tag{5.45}$$

$$+ \mathbb{E}\sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T, \tag{5.46}$$

where we calculate the four terms in the last equation as shown in below:

$$Eq.\ (5.43)$$

$$= \mathbb{E}\sum_{j=1}^{m} \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$+ \mathbb{E}\sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \mathbf{x}\mathbf{x}^T \mathbf{s}_{t_j}\mathbf{s}_{t_j}^T$$

$$= \sum_{k=1}^{d}\sum_{j=1}^{m} p_k \frac{1}{m^4 p_k^4} \mathbf{e}_k\mathbf{e}_k^T \mathbf{x}\mathbf{x}^T \mathbf{e}_k\mathbf{e}_k^T \mathbf{e}_k\mathbf{e}_k^T \mathbf{x}\mathbf{x}^T \mathbf{e}_k\mathbf{e}_k^T$$

$$+ \mathbb{E}\sum_{i\neq j\in[m]}\sum_{k=1}^{d}\sum_{q=1}^{d} p_k p_q \frac{1}{m^4 p_k^2 p_q^2} \mathbf{e}_k\mathbf{e}_k^T \mathbf{x}\mathbf{x}^T \mathbf{e}_k\mathbf{e}_k^T \mathbf{e}_q\mathbf{e}_q^T \mathbf{x}\mathbf{x}^T \mathbf{e}_q\mathbf{e}_q^T$$

$$= \sum_{k=1}^{d} \frac{x_k^4}{m^3 p_k^3}\mathbf{e}_k\mathbf{e}_k^T + \sum_{k=1}^{d} \frac{(m^2-m)x_k^4}{m^4 p_k^2}\mathbf{e}_k\mathbf{e}_k^T$$

$$= \sum_{k=1}^{d} (\frac{1}{m^3 p_k^3} + \frac{m-1}{m^3 p_k^2}) x_k^4 \mathbf{e}_k \mathbf{e}_k^T; \tag{5.47}$$

Eq. (5.44)

$$= \mathbb{E}\left[\sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \sum_{i\neq j\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\right]$$

$$= \mathbb{E}\sum_{i\neq j\neq g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T$$

$$+ \mathbb{E}\sum_{i\neq j, i=g, j\neq h, g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T$$

$$+ \mathbb{E}\sum_{i\neq j, i=h, j\neq g, g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T$$

$$+ \mathbb{E}\sum_{i\neq j, i\neq g, j=h, g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T$$

$$+ \mathbb{E}\sum_{i\neq j, i\neq h, j=g, g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T$$

$$+ \mathbb{E}\sum_{i\neq j, i=g, j=h, g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T$$

$$+ \mathbb{E}\sum_{i\neq j, i=h, j=g, g\neq h\in[m]} \mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_h}\mathbf{s}_{t_h}^T$$

$$= \sum_{k_1,k_2,k_3,k_4=1}^{d} \frac{m(m-1)(m-2)(m-3)}{m^4} x_{k_1}x_{k_2}x_{k_3}x_{k_4}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_3}\mathbf{e}_{k_4}^T$$

$$+ \sum_{k_1,k_2,k_4=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} x_{k_1}^2 x_{k_2}x_{k_4}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_1}\mathbf{e}_{k_4}^T$$

$$+ \sum_{k_1,k_2,k_3=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} x_{k_1}^2 x_{k_2}x_{k_3}\mathbf{e}_{k_1}\mathbf{e}_{k_2}^T\mathbf{e}_{k_3}\mathbf{e}_{k_1}^T$$

$$
\begin{aligned}
&+ \sum_{k_1,k_2,k_3=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_2}} x_{k_1} x_{k_2}^2 x_{k_3} \mathbf{e}_{k_1} \mathbf{e}_{k_2}^T \mathbf{e}_{k_3} \mathbf{e}_{k_2}^T \\
&+ \sum_{k_1,k_2,k_4=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_2}} x_{k_1} x_{k_2}^2 x_{k_4} \mathbf{e}_{k_1} \mathbf{e}_{k_2}^T \mathbf{e}_{k_2} \mathbf{e}_{k_4}^T \\
&+ \sum_{k_1,k_2}^{d} \frac{m(m-1)}{m^4 p_{k_1} p_{k_2}} x_{k_1}^2 x_{k_2}^2 \mathbf{e}_{k_1} \mathbf{e}_{k_2}^T \mathbf{e}_{k_1} \mathbf{e}_{k_2}^T \\
&+ \sum_{k_1,k_2=1}^{d} \frac{m(m-1)}{m^4 p_{k_1} p_{k_2}} x_{k_1}^2 x_{k_2}^2 \mathbf{e}_{k_1} \mathbf{e}_{k_2}^T \mathbf{e}_{k_2} \mathbf{e}_{k_1}^T \\
&= \sum_{k_2=1}^{d} x_{k_2}^2 \sum_{k_1,k_4=1}^{d} \frac{m(m-1)(m-2)(m-3)}{m^4} x_{k_1} x_{k_4} \mathbf{e}_{k_1} \mathbf{e}_{k_4}^T \\
&+ \sum_{k_1,k_4=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} x_{k_1}^3 x_{k_4} \mathbf{e}_{k_1} \mathbf{e}_{k_4}^T \\
&+ \sum_{k_2=1}^{d} x_{k_2}^2 \sum_{k_1=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} x_{k_1}^2 \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \\
&+ \sum_{k_1,k_2=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_2}} x_{k_1} x_{k_2}^3 \mathbf{e}_{k_1} \mathbf{e}_{k_2}^T \\
&+ \sum_{k_2=1}^{d} \frac{x_{k_2}^2}{p_{k_2}} \sum_{k_1,k_4=1}^{d} \frac{m(m-1)(m-2)}{m^4} x_{k_1} x_{k_4} \mathbf{e}_{k_1} \mathbf{e}_{k_4}^T \\
&+ \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^4 \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \\
&+ \sum_{k_2=1}^{d} \frac{x_{k_2}^2}{p_{k_2}} \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}} x_{k_1}^2 \mathbf{e}_{k_1} \mathbf{e}_{k_1}^T \\
&= \frac{\|\mathbf{x}\|_2^2 m(m-1)(m-2)(m-3)}{m^4} \mathbf{x}\mathbf{x}^T
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{m(m-1)(m-2)}{m^4}\mathbb{D}(\{\frac{x_k^2}{p_k}\})\mathbf{x}\mathbf{x}^T \\
&+\frac{\|\mathbf{x}\|_2^2 m(m-1)(m-2)}{m^4}\sum_{k=1}^{d}\frac{x_k^2}{p_k}\mathbf{e}_k\mathbf{e}_k^T \\
&+\frac{m(m-1)(m-2)}{m^4}\mathbf{x}\mathbf{x}^T\mathbb{D}(\{\frac{x_k^2}{p_k}\}) \\
&+\frac{m(m-1)(m-2)}{m^4}\sum_{k=1}^{d}\frac{x_k^2}{p_k}\mathbf{x}\mathbf{x}^T \\
&+\frac{m(m-1)}{m^4}\sum_{k=1}^{d}\frac{x_k^4}{p_k^2}\mathbf{e}_k\mathbf{e}_k^T \\
&+\frac{m(m-1)}{m^4}\sum_{k=1}^{d}\frac{x_k^2}{p_k}\sum_{k=1}^{d}\frac{x_k^2}{p_k}\mathbf{e}_k\mathbf{e}_k^T; \qquad (5.48)
\end{aligned}
$$

$$
\begin{aligned}
&Eq.\ (5.45) \\
=&\ \mathbb{E}\sum_{j=1}^{m}\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T\sum_{i\neq j\in[m]}\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \\
=&\ \mathbb{E}\sum_{g\neq i\neq j\in[m]}\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \\
&+\mathbb{E}\sum_{g=i\neq j\in[m]}\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \\
&+\mathbb{E}\sum_{g=j\neq i\in[m]}\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_g}\mathbf{s}_{t_g}^T\mathbf{s}_{t_i}\mathbf{s}_{t_i}^T\mathbf{x}\mathbf{x}^T\mathbf{s}_{t_j}\mathbf{s}_{t_j}^T \\
=&\sum_{k_1,k_2,k_3=1}^{d}\frac{m(m-1)(m-2)}{m^4 p_{k_1}}\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{x}\mathbf{x}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{e}_{k_2}\mathbf{e}_{k_2}^T\mathbf{x}\mathbf{x}^T\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T \\
&+\sum_{k_1,k_3=1}^{d}\frac{m(m-1)}{m^4 p_{k_1}^2}\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{x}\mathbf{x}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{e}_{k_1}\mathbf{e}_{k_1}^T\mathbf{x}\mathbf{x}^T\mathbf{e}_{k_3}\mathbf{e}_{k_3}^T
\end{aligned}
$$

$$
\begin{aligned}
&+ \sum_{k_1,k_2=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \mathbf{x}\mathbf{x}^T \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \mathbf{e}_{k_2}\mathbf{e}_{k_2}^T \mathbf{x}\mathbf{x}^T \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \\
&= \sum_{k_1,k_3=1}^{d} \frac{m(m-1)(m-2)}{m^4 p_{k_1}} x_{k_1}^3 x_{k_3} \mathbf{e}_{k_1}\mathbf{e}_{k_3}^T \\
&+ \sum_{k_1,k_3=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^3 x_{k_3} \mathbf{e}_{k_1}\mathbf{e}_{k_3}^T + \sum_{k_1=1}^{d} \frac{m(m-1)}{m^4 p_{k_1}^2} x_{k_1}^4 \mathbf{e}_{k_1}\mathbf{e}_{k_1}^T \\
&= \frac{m(m-1)(m-2)}{m^4} \mathbb{D}(\{\frac{x_k^2}{p_k}\}) \mathbf{x}\mathbf{x}^T \\
&+ \frac{m(m-1)}{m^4} \mathbb{D}(\{\frac{x_k^2}{p_k^2}\}) \mathbf{x}\mathbf{x}^T \\
&+ \frac{m(m-1)}{m^4} \sum_{k=1}^{d} \frac{x_k^4}{p_k^2} \mathbf{e}_k\mathbf{e}_k^T; \qquad (5.49)
\end{aligned}
$$

Eq. (5.46)

$$
\begin{aligned}
&= \frac{m(m-1)(m-2)}{m^4} \mathbf{x}\mathbf{x}^T \mathbb{D}(\{\frac{x_k^2}{p_k}\}) \\
&+ \frac{m(m-1)}{m^4} \mathbf{x}\mathbf{x}^T \mathbb{D}(\{\frac{x_k^2}{p_k^2}\}) \\
&+ \frac{m(m-1)}{m^4} \sum_{k=1}^{d} \frac{x_k^4}{p_k^2} \mathbf{e}_k\mathbf{e}_k^T. \qquad (5.50)
\end{aligned}
$$

Combing the above terms with simplification and reformulation completes the proof of Eq. (5.13). To this end, we complete the whole proof. □

5.6.3 Proof of Lemma 5.6

Proof. According to the setting, we have that

$$
\|\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T\|_2 \overset{(a)}{=} \|\mathbf{S}\mathbf{S}^T\mathbf{x}\|_2^2
$$

$$
\begin{aligned}
&= \| \sum_{j=1}^{m} \mathbf{s}_{t_j} \mathbf{s}_{t_j}^T \mathbf{x} \|_2^2 \\
&= \| \sum_{j=1}^{m} \frac{1}{mp_{t_j}} x_{t_j} \mathbf{e}_{t_j} \|_2^2 \\
&= \| \sum_{j=1}^{m} \sum_{k=1}^{d} \frac{\delta_{t_j k}}{mp_k} x_k \mathbf{e}_k \|_2^2 \\
&= \sum_{k=1}^{d} (\sum_{j=1}^{m} \frac{\delta_{t_j k} x_k}{mp_k})^2 \\
&\overset{(b)}{=} \sum_{k \in \Gamma} (\sum_{j=1}^{m} \frac{\delta_{t_j k} x_k}{mp_k})^2, \qquad (5.51)
\end{aligned}
$$

where we let $\Gamma = \{\gamma_t\}_{t=1}^{|\Gamma|}$ be a set containing at most m different elements of $[d]$ with its cardinality $|\Gamma| \leq m$.

In Eq. (5.51), (a) follows because $\mathbf{S}\mathbf{S}^T\mathbf{x}\mathbf{x}^T\mathbf{S}\mathbf{S}^T$ is a positive semidefinite matrix of rank 1, $\delta_{t_j k}$ returns 1 only when $t_j = k$ and 0 otherwise, and $\mathbb{P}(\delta_{t_j k} = 1) = \mathbb{P}(t_j = k) = p_k$. (b) holds due to that we perform random sampling with replacement m times on the d entries of $\mathbf{x} \in \mathbb{R}^d$, and consequently at most m certain different entries from $\mathbf{x}$ are sampled.

Let $k = \gamma_1$ with $\gamma_1 \in \Gamma$, and we first bound $|\sum_{j=1}^{m} \frac{\delta_{t_j \gamma_1} x_{\gamma_1}}{mp_{\gamma_1}}|$. Define a random variable $a_j = \frac{\delta_{t_j \gamma_1} x_{\gamma_1}}{mp_{\gamma_1}} - \frac{x_{\gamma_1}}{m}$ for all $j \in [m]$. We can easily check that $\{a_j\}_{j=1}^{m}$ are independent with $\mathbb{E}[a_j] = 0$, so that we can leverage Theorem 5.3 to continue our following analysis. We see that

$$
\max_{j \in [m]} |a_j| = \max\{\frac{|x_{\gamma_1}|}{m}(\frac{1}{p_{\gamma_1}} - 1), \frac{|x_{\gamma_1}|}{m}\} \leq \frac{|x_{\gamma_1}|}{mp_{\gamma_1}}, \qquad (5.52)
$$

and

$$\sum_{j=1}^{m} \mathbb{E}\left[a_j^2\right] = \frac{x_{\gamma_1}^2}{mp_{\gamma_1}} - \frac{x_{\gamma_1}^2}{m}. \tag{5.53}$$

Thus, applying Theorem 5.3 with $R = \frac{|x_{\gamma_1}|}{mp_{\gamma_1}}$ and $\sigma^2 = \frac{x_{\gamma_1}^2}{mp_{\gamma_1}} - \frac{x_{\gamma_1}^2}{m}$ obtains that

$$\mathbb{P}(|\sum_{j=1}^{m} a_j| \geq \epsilon) \leq 2\exp(\frac{-\epsilon^2/2}{x_{\gamma_1}^2/(mp_{\gamma_1}) - x_{\gamma_1}^2/m + |x_{\gamma_1}|\epsilon/(3mp_{\gamma_1})}), \tag{5.54}$$

whose RHS is denoted by η_{γ_1}. Then, with probability at least $1-\eta_{\gamma_1}$ we have $|\sum_{j=1}^{m} a_j| \leq \epsilon$, i.e., $|\sum_{j=1}^{m} \frac{\delta_{t_j\gamma_1} x_{\gamma_1}}{mp_1}| \leq |x_{\gamma_1}| + \epsilon$. We then replace ϵ by other variables to obtain that

$$\begin{aligned}|x_{\gamma_1}| + \epsilon &= |x_{\gamma_1}| \\ &+ \log(\frac{2}{\eta_{\gamma_1}})\left[\frac{|x_{\gamma_1}|}{3mp_{\gamma_1}} + |x_{\gamma_1}|\sqrt{\frac{1}{9m^2p_{\gamma_1}^2} + \frac{2}{\log(2/\eta_{\gamma_1})}(\frac{1}{mp_{\gamma_1}} - \frac{1}{m})}\right],\end{aligned} \tag{5.55}$$

which is denoted by $f(x_{\gamma_1}, \eta_{\gamma_1}, m)$.

In a similar way, we can bound $|\sum_{j=1}^{m} \frac{\delta_{t_jk} x_k}{mp_k}|$ for any other $k \in [d]$. The lemma then follows by using the union bound over cases for all $k \in [d]$. □

5.6.4 Proof of Theorem 5.1

Proof. We have to prove that the unbiased estimator for $\mathbf{C}$ is Eq. (5.1), i.e., $\mathbf{C}_e = \widehat{\mathbf{C}}_1 - \widehat{\mathbf{C}}_2$, where $\widehat{\mathbf{C}}_1 = \frac{m}{mn-n}\sum_{i=1}^{n} \mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$, and $\widehat{\mathbf{C}}_2 = \frac{m}{mn-n}\sum_{i=1}^{n} \mathbb{D}(\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T)\mathbb{D}(\mathbf{b}_i)$ with $b_{ki} = \frac{1}{1+(m-1)p_{ki}}$. Note that each $\mathbf{S}_i$ is created by running Algorithm 5.1, and

$\{\mathbf{S}_i\}_{i=1}^n$ are independent matrices. Thus, taking all summands $\mathbb{E}[\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T]$ together and leveraging Eq. (5.9) in Lemma 5.5 achieves the expectation of $\widehat{\mathbf{C}}_1$,

$$\begin{aligned}\mathbb{E}[\widehat{\mathbf{C}}_1] &= \frac{m}{nm-n}\mathbb{E}\sum_{i=1}^n \mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T \\ &= \frac{m}{nm-n}\sum_{i=1}^n\left[\sum_{k=1}^d \frac{x_{ki}^2}{mp_{ki}}\mathbf{e}_k\mathbf{e}_k^T + \frac{m-1}{m}\mathbf{x}_i\mathbf{x}_i^T\right] \\ &= \frac{1}{nm-n}\sum_{i=1}^n\sum_{k=1}^d \frac{x_{ki}^2}{p_{ki}}\mathbf{e}_k\mathbf{e}_k^T + \frac{1}{n}\mathbf{X}\mathbf{X}^T. \qquad (5.56)\end{aligned}$$

Eq. (5.56) indicates that $\widehat{\mathbf{C}}_1$ is a biased estimator for the original covariance matrix $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T$. We still need to apply a debiasing procedure to $\widehat{\mathbf{C}}_1$ to get an unbiased estimator. By Eq. (5.10) in Lemma 5.5, it can be shown that

$$\begin{aligned}\mathbb{E}[\widehat{\mathbf{C}}_2] &= \frac{m}{nm-n}\sum_{i=1}^n \mathbb{E}\left[\mathbb{D}(\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T)\right]\mathbb{D}(\mathbf{b}_i) \\ &= \frac{1}{nm-n}\sum_{i=1}^n\sum_{k=1}^d \frac{x_{ki}^2}{p_{ki}}\mathbf{e}_k\mathbf{e}_k^T. \qquad (5.57)\end{aligned}$$

Combing Eq. (5.56) with Eq. (5.57), we immediately see that $\mathbf{C}_e = \widehat{\mathbf{C}}_1 - \widehat{\mathbf{C}}_2$ is unbiased for $\mathbf{C}$. □

5.6.5 Proof of Theorem 5.2

Proof. Here, we have to bound the error $\|\mathbf{C}_e - \mathbf{C}\|_2$. To make the representation compact, we define $\mathbf{A}_i = \mathbf{A}_{i_1} - \mathbf{A}_{i_2} - \mathbf{A}_{i_3}$ with $\mathbf{A}_{i_1} = \frac{m\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T}{nm-n}$, $\mathbf{A}_{i_2} = \frac{m\mathbb{D}(\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T)\mathbb{D}(\mathbf{b}_i)}{nm-n}$, $\mathbf{A}_{i_3} = \frac{\mathbf{x}_i\mathbf{x}_i^T}{n}$. Then, $\sum_{i=1}^n \mathbf{A}_i = \mathbf{C}_e - \mathbf{C}$ holds. It is straightforward to see that $\{\mathbf{A}_i\}_{i=1}^n$ are independent zero-mean random matrices, which are exactly the setting of the Matrix Bernstein inequality, as shown in Theorem 5.3. To bound $\|\mathbf{C}_e - \mathbf{C}\|_2$ via Theorem 5.3, we need to calculate the relevant parameters R and σ^2 that characterize the range and variance of $\mathbf{A}_i$ respectively.

We first derive R by bounding $\|\mathbf{A}_i\|_2$ so that $\|\mathbf{A}_i\|_2 \leq R$ for all $i \in [n]$. Expanding $\|\mathbf{A}_i\|_2$ gets that

$$\begin{aligned}\|\mathbf{A}_i\|_2 &= \|\mathbf{A}_{i_1} - \mathbf{A}_{i_2} - \mathbf{A}_{i_3}\|_2 \\ &\leq \|\mathbf{A}_{i_1} - \mathbf{A}_{i_2}\|_2 + \|\mathbf{A}_{i_3}\|_2 \\ &\leq \|\mathbf{A}_{i_1}\|_2 + \|\mathbf{A}_{i_3}\|_2. \end{aligned} \tag{5.58}$$

The last inequality in Eq. (5.58) results from

$$\begin{aligned}\|\mathbf{A}_{i_1} - \mathbf{A}_{i_2}\|_2 &= \max_{k\in[d]} |\lambda_k(\mathbf{A}_{i_1} - \mathbf{A}_{i_2})| \\ &\overset{(a)}{\leq} \max\{|\lambda_d(\mathbf{A}_{i_1}) - \lambda_1(\mathbf{A}_{i_2})|, |\lambda_1(\mathbf{A}_{i_1}) - \lambda_d(\mathbf{A}_{i_2})|\} &(5.59)\\ &\overset{(b)}{=} \max\{\lambda_1(\mathbf{A}_{i_2}), |\lambda_1(\mathbf{A}_{i_1}) - \lambda_d(\mathbf{A}_{i_2})|\} &(5.60)\\ &\overset{(c)}{=} \max\{\lambda_1(\mathbf{A}_{i_2}), \lambda_1(\mathbf{A}_{i_1}) - \lambda_d(\mathbf{A}_{i_2})\} &(5.61)\\ &\overset{(d)}{\leq} \lambda_1(\mathbf{A}_{i_1}) &(5.62)\\ &\overset{(e)}{=} \|\mathbf{A}_{i_1}\|_2, &(5.63)\end{aligned}$$

where $\lambda_k(\cdot)$ is the k-th largest eigenvalue.

(a) follows from that $\lambda_k(\mathbf{A}_{i_1}) - \lambda_1(\mathbf{A}_{i_2}) \leq \lambda_k(\mathbf{A}_{i_1} - \mathbf{A}_{i_2}) \leq \lambda_k(\mathbf{A}_{i_1}) - \lambda_d(\mathbf{A}_{i_2})$ for any $k \in [d]$, which can be proved by combining Theorem 5.4 with the fact that $\lambda_d(-\mathbf{A}_{i_2}) = -\lambda_1(\mathbf{A}_{i_2})$ and $\lambda_1(-\mathbf{A}_{i_2}) = -\lambda_d(\mathbf{A}_{i_2})$ for $\mathbf{A}_{i_2} \in \mathbb{R}^{d\times d}$.

(b) holds because of that $\lambda_{k\geq 2}(\mathbf{A}_{i_1}) = 0$ since $\mathbf{A}_{i_1}$ is a positive semidefinite matrix of rank 1, and $\lambda_{k\in[d]}(\mathbf{A}_{i_2}) \geq 0$ since $\mathbf{A}_{i_2}$ is positive semidefinite.

(c) follows owing to that $\lambda_1(\mathbf{A}_{i_1}) = \mathrm{Tr}(\mathbf{A}_{i_1}) \geq \mathrm{Tr}(\mathbf{A}_{i_2}) = \sum_{k=1}^{d} \lambda_k(\mathbf{A}_{i_2}) \geq \lambda_d(\mathbf{A}_{i_2}) \geq 0$, where the first equality holds because $\lambda_{k\geq 2}(\mathbf{A}_{i_1}) = 0$, the first inequality results from the fact that the diagonal matrix $\mathbf{A}_{i_2}$ is constructed by the diagonal elements of $\mathbf{A}_{i_1}$ multiplied by positive scalars not bigger than 1, and the second inequality is the consequence of $\lambda_{k\in[d]}(\mathbf{A}_{i_2}) \geq 0$.

(d) results from that $\lambda_{k\in[d]}(\mathbf{A}_{i_2}) \geq 0$.

(e) follows owing to that $\mathbf{A}_{i_1}$ is positive semidefinite.

Now, we only need to bound $\|\mathbf{A}_{i_1}\|_2$ and $\|\mathbf{A}_{i_3}\|_2$. We have that

$$\|\mathbf{A}_{i_3}\|_2 = \|\frac{\mathbf{x}_i\mathbf{x}_i^T}{n}\|_2 = \frac{\|\mathbf{x}_i\|_2^2}{n}. \tag{5.64}$$

Then, Lemma 5.6 reveals that with probability at least $1 - \sum_{k=1}^{d} \eta_{ki}$,

$$\|\mathbf{A}_{i_1}\|_2 \leq \frac{m}{nm-n} \sum_{k\in\Gamma_i} f^2(x_{ki}, \eta_{ki}, m), \tag{5.65}$$

where $\Gamma_i = \{\gamma_{ti}\}_{t=1}^{|\Gamma_i|}$ is a set occupying at most m different elements of $[d]$ with its cardinality $|\Gamma_i| \leq m$, and $f(x_{ki}, \eta_{ki}, m) = |x_{ki}| + \log(\frac{2}{\eta_{ki}}) \left[\frac{|x_{ki}|}{3mp_{ki}} + |x_{ki}|\sqrt{\frac{1}{9m^2p_{ki}^2} + \frac{2}{\log(2/\eta_{ki})}(\frac{1}{mp_{ki}} - \frac{1}{m})}\right]$.

We derive the similar results for all $\{\mathbf{x}_i\}_{i=1}^n$. Then, by union bound, with probability at least $1 - \sum_{i=1}^{n}\sum_{k=1}^{d} \eta_{ki}$, we have

$$R = \max_{i\in[n]} \left[\frac{m}{nm-n} \sum_{k\in\Gamma_i} f^2(x_{ki}, \eta_{ki}, m) + \frac{\|\mathbf{x}_i\|_2^2}{n}\right]. \tag{5.66}$$

Applying the well known inequality $(\sum_{t=1}^{n} a_t)^2 \leq n\sum_{t=1}^{n} a_t^2$, we have

$$f^2(x_{ki}, \eta_{ki}, m) \leq 3x_{ki}^2 + 3\log^2(\frac{2}{\eta_{ki}})\frac{x_{ki}^2}{9m^2p_{ki}^2} + 3\log^2(\frac{2}{\eta_{ki}})\frac{x_{ki}^2}{9m^2p_{ki}^2}$$

$$
\begin{aligned}
&+ 6\log(\frac{2}{\eta_{ki}})(\frac{x_{ki}^2}{mp_{ki}} - \frac{x_{ki}^2}{m}) \\
&\leq 3x_{ki}^2 + \log^2(\frac{2}{\eta_{ki}})\frac{2x_{ki}^2}{3m^2p_{ki}^2} + \log(\frac{2}{\eta_{ki}})\frac{6x_{ki}^2}{mp_{ki}}.
\end{aligned}
\tag{5.67}
$$

Before continuing characterizing R in Eq. (5.66), we set the sampling probabilities as $p_{ki} = \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$. It is easy to check that $\sum_{k=1}^d p_{ki} = 1$. For $0 < \alpha < 1$, we also have $p_{ki} \geq \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}$, then plugging it in the second and third term of Eq. (5.67) respectively getting that

$$
f^2(x_{ki}, \eta_{ki}, m) \leq 3x_{ki}^2 + \log^2(\frac{2}{\eta_{ki}})\frac{2\|\mathbf{x}_i\|_1^2}{3m^2\alpha^2} + \log(\frac{2}{\eta_{ki}})\frac{6|\mathbf{x}_{ki}|\|\mathbf{x}_i\|_1}{m\alpha}.
\tag{5.68}
$$

Equipped with Eq. (5.66) and setting $\eta_{ki} = \frac{\eta}{nd}$ for all $i \in [n]$ and $k \in [d]$, we bound R with probability at least $1 - \sum_{i=1}^n\sum_{k=1}^d \eta_{ki} = 1 - \eta$ by

$$
\begin{aligned}
R &\leq \max_{i\in[n]}\left[\frac{m}{nm-n}\sum_{k\in\Gamma_i}\left(3x_{ki}^2 + \log^2(\frac{2nd}{\eta})\frac{2\|\mathbf{x}_i\|_1^2}{3m^2\alpha^2} + \log(\frac{2nd}{\eta})\frac{6|\mathbf{x}_{ki}|\|\mathbf{x}_i\|_1}{m\alpha}\right)\right. \\
&\quad \left. + \frac{\|\mathbf{x}_i\|_2^2}{n}\right] \\
&\leq \max_{i\in[n]}\left[\frac{2}{n}\left(3\|\mathbf{x}_i\|_2^2 + \log^2(\frac{2nd}{\eta})\frac{2\|\mathbf{x}_i\|_1^2}{3m\alpha^2} + \log(\frac{2nd}{\eta})\frac{6\|\mathbf{x}_i\|_1^2}{m\alpha}\right) + \frac{\|\mathbf{x}_i\|_2^2}{n}\right] \\
&\leq \max_{i\in[n]}\left[\frac{7\|\mathbf{x}_i\|_2^2}{n} + \log^2(\frac{2nd}{\eta})\frac{14\|\mathbf{x}_i\|_1^2}{nm\alpha^2}\right],
\end{aligned}
\tag{5.69}
$$

where the second inequality follows from that $\frac{m}{m-1} \leq 2$ for $m \geq 2$ and $|\Gamma_i| \leq m$, and the last inequality results from that $\alpha \leq 1$ and $\log(\frac{2nd}{\eta}) \geq 1$ for $n \geq 1$, $d \geq 2$, and $\eta \leq 1$.

At this stage, we have to derive σ^2 by only bounding for $\|\sum_{i=1}^n \mathbb{E}[\mathbf{A}_i\mathbf{A}_i]\|_2$ since $\mathbf{A}_i$ is symmetric. Expanding $\mathbb{E}[\mathbf{A}_i\mathbf{A}_i]$

obtains that

$$\mathbf{0} \preceq \mathbb{E}\left[\mathbf{A}_i\mathbf{A}_i\right] = \mathbb{E}\left[\mathbf{A}_{i_1}\mathbf{A}_{i_1} + \mathbf{A}_{i_2}\mathbf{A}_{i_2} + \mathbf{A}_{i_3}\mathbf{A}_{i_3} - \mathbf{A}_{i_1}\mathbf{A}_{i_2} - \mathbf{A}_{i_2}\mathbf{A}_{i_1} \right.$$
$$\left. -\mathbf{A}_{i_1}\mathbf{A}_{i_3} - \mathbf{A}_{i_3}\mathbf{A}_{i_1} + \mathbf{A}_{i_2}\mathbf{A}_{i_3} + \mathbf{A}_{i_3}\mathbf{A}_{i_2}\right],$$

in RHS of which, we bound the expectation of each term. Specifically, invoking Lemma 5.5, we have that

$$\begin{aligned}
& n^2\mathbb{E}\left[\mathbf{A}_i\mathbf{A}_i\right] \\
&= \underbrace{\sum_{k=1}^{d}\left[\frac{4}{m(m-1)p_{ki}^2} + \frac{1}{(m-1)^2 m p_{ki}^3}\right] x_{ki}^4 \mathbf{e}_k\mathbf{e}_k^T}_{①} \\
&\quad + \underbrace{\sum_{k=1}^{d}\left[\frac{\|\mathbf{x}_i\|_2^2(m-2)}{m(m-1)} + \frac{1}{m(m-1)}\sum_{k=1}^{d}\frac{x_{ki}^2}{p_{ki}}\right]\frac{x_{ki}^2}{p_{ki}}\mathbf{e}_k\mathbf{e}_k^T}_{②} \\
&\quad + \underbrace{\left[\frac{\|\mathbf{x}_i\|_2^2(m^2-5m+6)}{m(m-1)} + \frac{m-2}{m(m-1)}\sum_{k=1}^{d}\frac{x_{ki}^2}{p_{ki}}\right]\mathbf{x}_i\mathbf{x}_i^T}_{③} \\
&\quad + \underbrace{\frac{2(m-2)}{m(m-1)}\mathbf{x}_i\mathbf{x}_i^T\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}}\})}_{④} + \underbrace{\frac{1}{m(m-1)}\mathbf{x}_i\mathbf{x}_i^T\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}^2}\})}_{⑤} \\
&\quad + \underbrace{\frac{2(m-2)}{m(m-1)}\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}}\})\mathbf{x}_i\mathbf{x}_i^T}_{⑥} + \underbrace{\frac{1}{m(m-1)}\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}^2}\})\mathbf{x}_i\mathbf{x}_i^T}_{⑦} \\
&\quad + \underbrace{\mathbb{D}(\mathbf{b}_i)\mathbb{D}(\mathbf{b}_i)\sum_{k=1}^{d}\left[\frac{1}{m(m-1)^2p_{ki}^3} + \frac{7}{m(m-1)p_{ki}^2}\right.}_{⑧} \\
&\quad \underbrace{\left. + \frac{6(m-2)}{m(m-1)p_{ki}} + \frac{(m-2)(m-3)}{m(m-1)}\right] x_{ki}^4\mathbf{e}_k\mathbf{e}_k^T}_{⑧}
\end{aligned}$$

$$+\underbrace{\|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T}_{⑨}+\underbrace{\sum_{k=1}^{d}(\frac{1}{(m-1)p_{ki}}+1)x_{ki}^2\mathbf{e}_k\mathbf{e}_k^T\mathbb{D}(\mathbf{b}_i)\mathbf{x}_i\mathbf{x}_i^T}_{⑩}$$

$$+\underbrace{\mathbf{x}_i\mathbf{x}_i^T\sum_{k=1}^{d}(\frac{1}{(m-1)p_{ki}}+1)x_{ki}^2\mathbf{e}_k\mathbf{e}_k^T\mathbb{D}(\mathbf{b}_i)}_{⑪}$$

$$-\underbrace{2\sum_{k=1}^{d}\left[\frac{1}{m(m-1)^2p_{ki}^3}+\frac{6}{m(m-1)p_{ki}^2}+\frac{3(m-2)}{m(m-1)p_{ki}}\right]x_{ki}^4\mathbf{e}_k\mathbf{e}_k^T\mathbb{D}(\mathbf{b}_i)}_{⑫}$$

$$-\underbrace{\frac{3(m-2)}{m(m-1)}\mathbf{x}_i\mathbf{x}_i^T\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}}\})\mathbb{D}(\mathbf{b}_i)}_{⑬}$$

$$-\underbrace{\frac{(m-2)(m-3)}{m(m-1)}\mathbf{x}_i\mathbf{x}_i^T\mathbb{D}(\{x_{ki}^2\})\mathbb{D}(\mathbf{b}_i)}_{⑭}$$

$$-\underbrace{\frac{3(m-2)}{m(m-1)}\mathbb{D}(\mathbf{b}_i)\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}}\})\mathbf{x}_i\mathbf{x}_i^T}_{⑮}$$

$$-\underbrace{\frac{(m-2)(m-3)}{m(m-1)}\mathbb{D}(\mathbf{b}_i)\mathbb{D}(\{x_{ki}^2\})\mathbf{x}_i\mathbf{x}_i^T}_{⑯}$$

$$-\underbrace{\sum_{k=1}^{d}\frac{x_{ki}^2}{(m-1)p_{ki}}\mathbf{e}_k\mathbf{e}_k^T\mathbf{x}_i\mathbf{x}_i^T}_{⑰}-\underbrace{\|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T}_{⑱}$$

$$-\underbrace{\sum_{k=1}^{d}\frac{x_{ki}^2}{(m-1)p_{ki}}\mathbf{x}_i\mathbf{x}_i^T\mathbf{e}_k\mathbf{e}_k^T}_{⑲}-\underbrace{\|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T}_{⑳}$$

$$\underbrace{-\frac{1}{m(m-1)}\mathbf{x}_i\mathbf{x}_i^T\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}^2}\})\mathbb{D}(\mathbf{b}_i)}_{⑳①}$$

$$\underbrace{-\frac{1}{m(m-1)}\mathbb{D}(\mathbf{b}_i)\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}^2}\})\mathbf{x}_i\mathbf{x}_i^T}_{㉒}. \tag{5.70}$$

Because of the limited space, $\mathbb{D}(\{\frac{x_{ki}^2}{p_{ki}}\})$ is to denote a square diagonal matrix in $\mathbb{R}^{d\times d}$ with $\{\frac{x_{ki}^2}{p_{ki}}\}_{k=1}^d$ on its diagonal, which is also extended to other similar notations.

In Eq. (5.70), it can be checked that for $m \geq 2$, we have

$$\begin{aligned}
⑩ - ⑰ &= \mathbf{0};\\
⑪ - ⑲ &= \mathbf{0};\\
④ - ⑬ + ⑤ - ⑭ - ㉑ &= \frac{\mathbf{x}_i\mathbf{x}_i^T}{m(m-1)}\mathbb{D}(\{\frac{((m-1)/p_{ki})x_{ki}^2}{1+(m-1)p_{ki}}\\
&\quad + \frac{(m-2)(m+1-1/p_{ki})x_{ki}^2}{1+(m-1)p_{ki}}\});\\
⑥ - ⑮ + ⑦ - ⑯ - ㉒ &= \mathbb{D}(\frac{((m-1)/p_{ki})x_{ki}^2}{1+(m-1)p_{ki}})\frac{\mathbf{x}_i\mathbf{x}_i^T}{m(m-1)}\\
&\quad + \mathbb{D}(\frac{(m-2)(m+1-1/p_{ki})x_{ki}^2}{1+(m-1)p_{ki}})\frac{\mathbf{x}_i\mathbf{x}_i^T}{m(m-1)};\\
③ + ⑨ - ⑱ - ⑳ &= \left[\frac{(6-4m)\|\mathbf{x}_i\|_2^2}{m(m-1)} + \frac{m-2}{m(m-1)}\sum_{k=1}^d \frac{x_{ki}^2}{p_{ki}}\right]\mathbf{x}_i\mathbf{x}_i^T\\
&\preceq \frac{1}{m}\sum_{k=1}^d \frac{x_{ki}^2}{p_{ki}}\mathbf{x}_i\mathbf{x}_i^T;\\
⑧ - ⑫ &\preceq \mathbf{0};\\
① &\preceq \sum_{k=1}^d \left[\frac{8x_{ki}^4}{m^2p_{ki}^2} + \frac{4x_{ki}^4}{m^3p_{ki}^3}\right]\mathbf{e}_k\mathbf{e}_k^T;
\end{aligned}$$

$$\textcircled{2} \preceq \sum_{k=1}^{d} \left[\frac{\|\mathbf{x}_i\|_2^2 x_{ki}^2}{m p_{ki}} + \frac{2x_{ki}^2}{m^2 p_{ki}} \sum_{k=1}^{d} \frac{x_{ki}^2}{p_{ki}} \right] \mathbf{e}_k \mathbf{e}_k^T. \tag{5.71}$$

Then, applying Eq. (5.70) and Eq. (5.71) obtains that

$$\begin{aligned}
&\mathbf{0} \preceq \mathbb{E}\left[\mathbf{A}_i \mathbf{A}_i\right] \\
&\preceq \frac{1}{n^2} \sum_{k=1}^{d} \left[\frac{8x_{ki}^4}{m^2 p_{ki}^2} + \frac{4x_{ki}^4}{m^3 p_{ki}^3} + \frac{\|\mathbf{x}_i\|_2^2 x_{ki}^2}{m p_{ki}} + \frac{2x_{ki}^2}{m^2 p_{ki}} \sum_{k=1}^{d} \frac{x_{ki}^2}{p_{ki}} \right] \mathbf{e}_k \mathbf{e}_k^T \\
&+ \frac{\mathbf{x}_i \mathbf{x}_i^T}{n^2 m(m-1)} \mathbb{D}(\{ \frac{((m-1)/p_{ki}) x_{ki}^2}{1+(m-1)p_{ki}} + \frac{(m-2)(m+1-1/p_{ki}) x_{ki}^2}{1+(m-1)p_{ki}} \}) \\
&+ \mathbb{D}(\{ \frac{((m-1)/p_{ki}) x_{ki}^2}{1+(m-1)p_{ki}} + \frac{(m-2)(m+1-1/p_{ki}) x_{ki}^2}{1+(m-1)p_{ki}} \}) \frac{\mathbf{x}_i \mathbf{x}_i^T}{n^2 m(m-1)} \\
&+ \frac{1}{n^2 m} \sum_{k=1}^{d} \frac{x_{ki}^2}{p_{ki}} \mathbf{x}_i \mathbf{x}_i^T.
\end{aligned} \tag{5.72}$$

With Eq. (5.72) in hand, we can formulate σ^2 as

$$\begin{aligned}
\sigma^2 &= \| \sum_{i=1}^{n} \mathbb{E}\left[\mathbf{A}_i \mathbf{A}_i\right] \|_2 \\
&\leq \sum_{i=1}^{n} \max_{k \in [d]} \frac{1}{n^2} \left[\frac{8x_{ki}^4}{m^2 p_{ki}^2} + \frac{4x_{ki}^4}{m^3 p_{ki}^3} + \frac{\|\mathbf{x}_i\|_2^2 x_{ki}^2}{m p_{ki}} + \frac{2x_{ki}^2}{m^2 p_{ki}} \sum_{k=1}^{d} \frac{x_{ki}^2}{p_{ki}} \right] \\
&\quad + \sum_{i=1}^{n} \max_{k \in [d]} \frac{1}{n^2} \left[\frac{2\|\mathbf{x}_i\|_2^2}{m(m-1)} (\frac{((m-1)/p_{ki}) x_{ki}^2}{1+(m-1)p_{ki}} \right. \\
&\quad \left. + \frac{(m-2)(m+1+1/p_{ki}) x_{ki}^2}{1+(m-1)p_{ki}}) \right] \\
&\quad + \frac{1}{n^2 m} \| \sum_{i=1}^{n} \sum_{k=1}^{d} \frac{x_{ki}^2}{p_{ki}} \mathbf{x}_i \mathbf{x}_i^T \|_2 \\
&\leq \sum_{i=1}^{n} \max_{k \in [d]} \frac{1}{n^2} \left[\frac{8x_{ki}^4}{m^2 p_{ki}^2} + \frac{4x_{ki}^4}{m^3 p_{ki}^3} + \frac{\|\mathbf{x}_i\|_2^2 x_{ki}^2}{m p_{ki}} + \frac{2x_{ki}^2}{m^2 p_{ki}} \sum_{k=1}^{d} \frac{x_{ki}^2}{p_{ki}} \right]
\end{aligned}$$

$$+\sum_{i=1}^{n}\max_{k\in[d]}\frac{1}{n^2}\left[\frac{8\|\mathbf{x}_i\|_2^2 x_{ki}^2}{mp_{ki}}\right]+\frac{1}{n^2m}\|\sum_{i=1}^{n}\sum_{k=1}^{d}\frac{x_{ki}^2}{p_{ki}}\mathbf{x}_i\mathbf{x}_i^T\|_2. \tag{5.73}$$

Again, we have to consider the sampling distributions $p_{ki}=\alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}+(1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$ with $0<\alpha<1$. Plugging $p_{ki}\geq\alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}$ and $p_{ki}\geq(1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$ in Eq. (5.73), we have

$$\begin{aligned}\sigma^2\leq&\sum_{i=1}^{n}\max_{k\in[d]}\frac{1}{n^2}\left[\frac{8\|\mathbf{x}_i\|_2^4}{m^2(1-\alpha)^2}+\frac{4\|\mathbf{x}_i\|_1^2\|\mathbf{x}_i\|_2^2}{m^3\alpha^2(1-\alpha)}+\frac{\|\mathbf{x}_i\|_2^4}{m(1-\alpha)}\right.\\&\left.+\frac{2\|\mathbf{x}_i\|_2^2}{m^2(1-\alpha)}\sum_{k=1}^{d}\frac{|x_{ki}|\|\mathbf{x}_i\|_1}{\alpha}\right]\\&+\sum_{i=1}^{n}\max_{k\in[d]}\frac{1}{n^2}\left[\frac{8\|\mathbf{x}_i\|_2^4}{m(1-\alpha)}\right]+\frac{1}{n^2m}\|\sum_{i=1}^{n}\sum_{k=1}^{d}\frac{|x_{ki}|\|\mathbf{x}_i\|_1}{\alpha}\mathbf{x}_i\mathbf{x}_i^T\|_2\\=&\sum_{i=1}^{n}\left[\frac{8\|\mathbf{x}_i\|_2^4}{n^2m^2(1-\alpha)^2}+\frac{4\|\mathbf{x}_i\|_1^2\|\mathbf{x}_i\|_2^2}{n^2m^3\alpha^2(1-\alpha)}+\frac{9\|\mathbf{x}_i\|_2^4}{n^2m(1-\alpha)}\right.\\&\left.+\frac{2\|\mathbf{x}_i\|_2^2\|\mathbf{x}_i\|_1^2}{n^2m^2\alpha(1-\alpha)}\right]+\|\sum_{i=1}^{n}\frac{\|\mathbf{x}_i\|_1^2\mathbf{x}_i\mathbf{x}_i^2}{n^2m\alpha}\|_2.\end{aligned} \tag{5.74}$$

Note that employing $p_{ki}=\Omega(\frac{|x_{ki}|^{4/3}}{\sum_{k=1}^{d}|x_{ki}|^{4/3}})$ for the term $\frac{4x_{ki}^4}{m^3p_{ki}^3}$ in Eq. (5.73) can produce a result tighter than that in Eq. (5.74), which is because of the fact that $(\sum_{k=1}^{d}|x_{ki}|^{4/3})^3\leq\|\mathbf{x}_i\|_1^2\|\mathbf{x}_i\|_2^2$ always holds owing to the Holder's inequality. However, it is not necessary to apply $p_{ki}=\Omega(\frac{|x_{ki}|^{4/3}}{\sum_{k=1}^{d}|x_{ki}|^{4/3}})$ to the term $\frac{4x_{ki}^4}{m^3p_{ki}^3}$ in Eq. (5.73), because the term $\frac{4\|\mathbf{x}_i\|_1^2\|\mathbf{x}_i\|_2^2}{n^2m^3\alpha^2(1-\alpha)}=O(\frac{\|\mathbf{x}_i\|_1^2\|\mathbf{x}_i\|_2^2}{n^2m^3})$ in Eq. (5.74) obtained by applying $p_{ki}=\alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}+(1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}=\Omega(\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}+\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2})$ to the term $\frac{4x_{ki}^4}{m^3p_{ki}^3}$ in Eq. (5.73) has already been small enough, which can be smaller than other terms in

Eq. (5.74) like $\frac{2\|\mathbf{x}_i\|_2^2\|\mathbf{x}_i\|_1^2}{n^2m^2\alpha(1-\alpha)} = O(\frac{\|\mathbf{x}_i\|_1^2\|\mathbf{x}_i\|_2^2}{n^2m^2})$. Similarly, applying other sampling probabilities $p_{ki} = \Omega(\frac{|x_{ki}|^q}{\sum_{k=1}^d |x_{ki}|^q})$ with $q \neq 1, \frac{4}{3}, 2$ to Eq. (5.73) will produce a result larger than Eq. (5.74), which may not be bounded. This is also why we only use $p_{ki} = \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2} = \Omega(\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1})$ to tighten R in Eq. (5.69). This derivation justifies our selection of $q = 1, 2$ in $p_{ki} = \Omega(\frac{|x_{ki}|^q}{\sum_{k=1}^d |x_{ki}|^q})$ used for constructing the sampling probability $p_{ki} = \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$.

We then invoke Theorem 5.3 to obtain that for $\epsilon \geq 0$,

$$\mathbb{P}(\|\mathbf{C}_e - \mathbf{C}\|_2 \geq \epsilon) \leq 2d\exp(\frac{-\epsilon^2/2}{\sigma^2 + R\epsilon/3}). \tag{5.75}$$

Denote the RHS of Eq. (5.75) by $\delta = 2d\exp(\frac{-\epsilon^2/2}{\sigma^2+R\epsilon/3})$ and consider the failure probability η in Eq. (5.69), then by union bound we have $\|\mathbf{C}_e - \mathbf{C}\|_2 \leq \epsilon$ holds with probability at least $1-\eta-\delta$. Furthermore, $\delta = 2d\exp(\frac{-\epsilon^2/2}{\sigma^2+R\epsilon/3})$ yields the following quadratic equation in ϵ

$$\frac{\epsilon^2}{2\log(2d/\delta)} - \frac{R\epsilon}{3} - \sigma^2 = 0. \tag{5.76}$$

Solving Eq. (5.76) gets only one positive root

$$\begin{aligned}\epsilon &= \log(\frac{2d}{\delta})\left[\frac{R}{3} + \sqrt{(\frac{R}{3})^2 + \frac{2\sigma^2}{\log(2d/\delta)}}\right] \\ &\leq \log(\frac{2d}{\delta})\frac{2R}{3} + \sqrt{2\sigma^2\log(\frac{2d}{\delta})}. \end{aligned} \tag{5.77}$$

Thus, immediately we have $\|\mathbf{C}_e - \mathbf{C}\|_2 \leq \log(\frac{2d}{\delta})\frac{2R}{3} + \sqrt{2\sigma^2\log(\frac{2d}{\delta})}$ holds with probability at least $1-\eta-\delta$, which completes the whole proof. $\square$

5.6.6 Proof of Corollary 5.1

Proof. According to the setting, substituting that $\|\mathbf{x}_i\|_2 \leq \tau$ for all $i \in [n]$, $\frac{\|\mathbf{x}_i\|_1}{\|\mathbf{x}_i\|_2} \leq \varphi$ with $1 \leq \varphi \leq \sqrt{d}$, and $m < d$ into Theorem 5.2 establishes that

$$\begin{aligned}
&\|\mathbf{C}_e - \mathbf{C}\|_2 \\
&\leq \widetilde{O}\Big(\frac{\tau^2}{n} + \frac{\tau^2\varphi^2}{nm} + \sqrt{\frac{\tau^4}{nm^2} + \frac{\tau^4\varphi^2}{nm^3} + \frac{\tau^4}{nm} + \frac{\tau^4\varphi^2}{nm^2} + \frac{\|\mathbf{C}\|_2\tau^2\varphi^2}{nm}}\Big) \\
&\leq \widetilde{O}\Big(\frac{\tau^2}{n} + \frac{\tau^2\varphi^2}{nm} + \frac{\tau^2\varphi}{m}\sqrt{\frac{1}{n}} + \tau^2\sqrt{\frac{1}{nm}} + \tau\varphi\sqrt{\frac{\|\mathbf{C}\|_2}{nm}}\Big), \qquad (5.78)
\end{aligned}$$

where the first inequality invokes $\sum_{i=n}^{n} \|\mathbf{x}_i\|_2^4 \leq n\tau^4$, and $\mathbf{C} = \sum_{i=1}^{n} \frac{\mathbf{x}_i\mathbf{x}_i^T}{n}$ is the original covariance matrix.

Also, we can adopt $\sum_{i=1}^{n} \|\mathbf{x}_i\|_2^4 \leq nd\tau^2\|\mathbf{C}\|_2$, which holds because $\sum_{i=1}^{n} \|\mathbf{x}_i\|_2^4 \leq \tau^2 \sum_{i=1}^{n} \|\mathbf{x}_i\|_2^2$ and $\sum_{i=1}^{n} \|\mathbf{x}_i\|_2^2 = n\mathrm{Tr}(\mathbf{C}) \leq nd\|\mathbf{C}\|_2$. □

Hence, we have

$$\begin{aligned}
&\|\mathbf{C}_e - \mathbf{C}\|_2 \\
&\leq \widetilde{O}\Big(\frac{\tau^2}{n} + \frac{\tau^2\varphi^2}{nm} + \tau\sqrt{\|\mathbf{C}\|_2}\sqrt{\frac{d}{nm^2} + \frac{d\varphi^2}{nm^3} + \frac{d}{nm} + \frac{d\varphi^2}{nm^2} + \frac{\varphi^2}{nm}}\Big) \\
&\leq \widetilde{O}\Big(\frac{\tau^2}{n} + \frac{\tau^2\varphi^2}{nm} + \frac{\tau\varphi}{m}\sqrt{\frac{d\|\mathbf{C}\|_2}{n}} + \tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \tau\varphi\sqrt{\frac{\|\mathbf{C}\|_2}{nm}}\Big). \qquad (5.79)
\end{aligned}$$

Finally, assigning $\|\mathbf{C}_e - \mathbf{C}\|_2$ by the smaller one of Eq. (5.78) and Eq. (5.79) completes the proof.

5.6.7 Proof of Corollary 5.2

Proof. The proof follows [16, Corollaries 4-6], where the key component $\|\mathbf{C}_e - \mathbf{C}_p\|_2$ is upper bounded by $\|\mathbf{C}_e - \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^T\|_2 +$

$\|\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}_p\|_2$. Then, the derivation results from Theorem 5.2 in this chapter and the Gaussian tail bounds in [16, Proposition 14].

[16, Proposition 14] shows that with probability at least $1-\zeta$ for $d \geq 2$,

$$\max_{i\in[n]} \|\mathbf{x}_i\|_2 \leq \sqrt{2\mathrm{Tr}(\mathbf{C}_p)\log(nd/\zeta)};$$
$$\|\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}_p\|_2 \leq O\big(\|\mathbf{C}_p\|_2\sqrt{\log(2/\zeta)/n}\big). \tag{5.80}$$

Then, applying them and Corollary 5.1 along with the fact that $\|\mathbf{x}_i\|_1 \leq \sqrt{d}\|\mathbf{x}_i\|_2$ and $\mathrm{Tr}(\mathbf{C}_p) \leq d\|\mathbf{C}_p\|_2$ establishes

$$\begin{aligned}
&\|\mathbf{C}_e - \mathbf{C}_p\|_2 \\
&\leq \|\mathbf{C}_e - \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T\|_2 + \|\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}_p\|_2 \\
&\leq \widetilde{O}\Big(\frac{\tau^2}{n} + \frac{\tau^2\varphi^2}{nm} + \frac{\tau^2\varphi}{m}\sqrt{\frac{1}{n}} + \tau^2\sqrt{\frac{1}{nm}} + \tau\varphi\sqrt{\frac{\|\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T\|_2}{nm}}\Big) \\
&\quad + \widetilde{O}\Big(\|\mathbf{C}_p\|_2\sqrt{\frac{1}{n}}\Big) \\
&\leq \widetilde{O}\Big(\frac{\tau^2}{n} + \frac{\tau^2\varphi^2}{nm} + \frac{\tau^2\varphi}{m}\sqrt{\frac{1}{n}} + \tau^2\sqrt{\frac{1}{nm}} + \tau\varphi\sqrt{\frac{\|\mathbf{C}_p\|_2}{nm}}\Big) \\
&\quad + \widetilde{O}\Big(\|\mathbf{C}_p\|_2\sqrt{\frac{1}{n}}\Big) &(5.81)\\
&\leq \widetilde{O}\Big(\frac{d^2\|\mathbf{C}_p\|_2}{nm} + \frac{d\|\mathbf{C}_p\|_2}{m}\sqrt{\frac{d}{n}} + d\|\mathbf{C}_p\|_2\sqrt{\frac{1}{nm}} + d\|\mathbf{C}_p\|_2\sqrt{\frac{1}{nm}} \\
&\quad + \|\mathbf{C}_p\|_2\sqrt{\frac{1}{n}}\Big) \\
&\leq \widetilde{O}\Big(\frac{d^2\|\mathbf{C}_p\|_2}{nm} + \frac{d\|\mathbf{C}_p\|_2}{m}\sqrt{\frac{d}{n}}\Big) &(5.82)
\end{aligned}$$

with probability at least $1-\eta-\delta-\zeta$, where Eq. (5.81) results from that we invoke Eq. (5.80) to get $\|\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T\|_2 \leq \|\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}_p\|_2 + \|\mathbf{C}_p\|_2 \leq \widetilde{O}(\|\mathbf{C}_p\|_2)$.

The proof for the low-rank case where rank($\mathbf{C}_p$)$\leq r$ additionally adopts

$$\begin{aligned}\|[\mathbf{C}_e]_r - \mathbf{C}_p\|_2 &\leq \|[\mathbf{C}_e]_r - \mathbf{C}_e\|_2 + \|\mathbf{C}_e - \mathbf{C}_p\|_2 \\ &\leq \|[\mathbf{C}_p]_r - \mathbf{C}_e\|_2 + \|\mathbf{C}_e - \mathbf{C}_p\|_2 \\ &\leq \|[\mathbf{C}_p]_r - \mathbf{C}_p\|_2 + \|\mathbf{C}_p - \mathbf{C}_e\|_2 + \|\mathbf{C}_e - \mathbf{C}_p\|_2 \\ &= 2\|\mathbf{C}_e - \mathbf{C}_p\|_2, \end{aligned} \tag{5.83}$$

where the last equality holds because $\text{rank}(\mathbf{C}_p) \leq r$. Then, armed with $\text{Tr}(\mathbf{C}_p) \leq \text{rank}(\mathbf{C}_p)\|\mathbf{C}_p\|_2 \leq r\|\mathbf{C}_p\|_2$, we have

$$\begin{aligned}&\|[\mathbf{C}_e]_r - \mathbf{C}_p\|_2 \\ &\leq O(\|\mathbf{C}_e - \mathbf{C}_p\|_2) \\ &\leq O(\|\mathbf{C}_e - \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T\|_2 + \|\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}_p\|_2) \\ &\leq \widetilde{O}\Big(\frac{rd\|\mathbf{C}_p\|_2}{nm} + \frac{r\|\mathbf{C}_p\|_2}{m}\sqrt{\frac{d}{n}} + r\|\mathbf{C}_p\|_2\sqrt{\frac{1}{nm}} + \|\mathbf{C}_p\|_2\sqrt{\frac{rd}{nm}} \\ &\quad + \|\mathbf{C}_p\|_2\sqrt{\frac{1}{n}}\Big) \\ &\leq \widetilde{O}\Big(\frac{rd\|\mathbf{C}_p\|_2}{nm} + \frac{r\|\mathbf{C}_p\|_2}{m}\sqrt{\frac{d}{n}} + \|\mathbf{C}_p\|_2\sqrt{\frac{rd}{nm}}\Big) \end{aligned} \tag{5.84}$$

with probability at least $1-\eta-\delta-\zeta$. □

Note that the result of [16, Corollary 5] is incorrect, since retaining the largest r eigenvalues and associated eigenvectors of a matrix that is not positive-(semi-)definite therein usually cannot obtain the best r-rank matrix approximation in the matrix spectral norm, which is also inconsistent with its proof.

5.6.8 Proof of Corollary 5.3

Proof. The given definitions also implicitly indicate that $\mathbf{C}_p$ and $\mathbf{C}_e$ are symmetric. Then, following [16], the desired bound in Corollary 5.3 immediately results from Corollary 5.2 combined with the Davis-Kahan Theorem [49] that shows $\|\widehat{\prod}_k - \prod_k\|_2 \leq \frac{1}{\lambda_k - \lambda_{k+1}}\|\mathbf{C}_e - \mathbf{C}_p\|_2$. □

5.7 Details for Counterparts

5.7.1 Theorems for Gauss-Inverse and UniSample-HD

We first use our notations to rephrase current theoretical results provided in [16, Theorem 3] and [10, Theorem 6], which correspond to *Gauss-Inverse* and *UniSample-HD*, respectively.

Theorem 5.7 ([16, Theorem 3])**.** *Let $d \geq 2$ and define*

$$S_1 = \|\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i\|_2^2\mathbf{x}_i\mathbf{x}_i^T\|_2, S_2 = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i\|_2^4.$$

There exists universal constants κ_1, $\kappa_2 > 0$ such that for any $0 < \delta < 1$, with probability at least $1 - \delta$,

$$\begin{aligned}&\|\mathbf{C}_e - \mathbf{C}\|_2 \\ &\leq \kappa_1\Big(\sqrt{\frac{d}{m}S_1} + \sqrt{\frac{d}{m^2}S_2}\Big)\sqrt{\frac{\log(d/\delta)}{n}} \\ &\quad + \kappa_2\frac{d\max_{i\in[n]}\|\mathbf{x}_i\|_2^2}{nm}\log(d/\delta). \qquad (5.85)\end{aligned}$$

Theorem 5.8 ([10, Theorem 6])**.** *Let each column of $\mathbf{S}_i \in \mathbb{R}^{d\times m}$ be chosen uniformly at random from the set of all canonical basis vectors without replacement. Let $\rho > 0$ be a bound such that*

$\|\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\|_2^2 \leq \rho\|\mathbf{x}_i\|_2^2$ *for all* $i \in [n]$. *Then, with probability at least* $1-\delta$

$$\|\mathbf{C}_e - \mathbf{C}\|_2 \leq \epsilon, \tag{5.86}$$

where $\delta = d\exp\left(\frac{-\epsilon^2/2}{\sigma^2+R\epsilon/3}\right)$, $R = \frac{1}{n}\Big[\Big(\frac{d(d-1)}{m(m-1)}\rho+1\Big)\max_{i\in[n]}\|\mathbf{x}_i\|_2^2$ $+\frac{d(d-m)}{m(m-1)}\max_{k\in[d],i\in[n]}x_{ki}^2\Big]$, *and* $\sigma^2 = \frac{d(d-1)}{nm(m-1)}\Big[\frac{(d-m)^2\max_{k\in[d]}\sum_{i=1}^n x_{ki}^4}{n(d-1)(m-1)}$ $+(\rho-\frac{m(m-1)}{d(d-1)})\max_{i\in[n]}\|\mathbf{x}_i\|_2^2\|\mathbf{C}\|_2 + \frac{d-m}{m-1}\rho\max_{i\in[n]}\|\mathbf{x}_i\|_2^2\|\mathbb{D}(\mathbf{C})\|_2$ $+\frac{2(d-m)\|\mathbf{X}\|_F^2}{n(m-1)}\max_{k\in[d],i\in[n]}x_{ki}^2\Big]$.

5.7.2 Discussion

In this subsection, we will simplify the foregoing two theorems by making Eq. (5.85) and Eq. (5.86) explicitly dependent on n, m and d. Our derivations are natural and straightforward, and we will not deliberately loose Eq. (5.85) and Eq. (5.86) in order to demonstrate the superiority of the theoretical results gained by our weighted sampling method. We define that $\max_{i\in n}\|\mathbf{x}_i\|_2 \leq \tau$.

In terms of Eq. (5.85) in Theorem 5.7, $S_1 \leq \max_{i\in[n]}\|\mathbf{x}_i\|_2^2\|\mathbf{C}\|_2$ and $S_2 \leq \max_{i\in[n]}\|\mathbf{x}_i\|_2^4$. Note that $\frac{1}{nd}\|\mathbf{X}\|_F^2 \leq \|\mathbf{C}\|_2 \leq \max_{i\in[n]}\|\mathbf{x}_i\|_2^2$. Then, Eq. (5.85) can be simplified and reformulated as

$$\begin{aligned}
&\|\mathbf{C}_e - \mathbf{C}\|_2 \\
&\leq \widetilde{O}\Big(\sqrt{\frac{d\|\mathbf{C}\|_2\max_{i\in[n]}\|\mathbf{x}_i\|_2^2}{nm}} + \sqrt{\frac{d\max_{i\in[n]}\|\mathbf{x}_i\|_2^4}{nm^2}} \\
&\quad + \frac{d\max_{i\in[n]}\|\mathbf{x}_i\|_2^2}{nm}\Big) \\
&\leq \widetilde{O}\Big(\tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2}{m}\sqrt{\frac{d}{n}} + \frac{\tau^2 d}{nm}\Big).
\end{aligned} \tag{5.87}$$

If applying $S_2 \leq d \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \|\mathbf{C}\|_2$ in the original paper [16], we will get that

$$\begin{aligned}
&\|\mathbf{C}_e - \mathbf{C}\|_2 \\
&\leq \widetilde{O}\Big(\sqrt{\frac{d\|\mathbf{C}\|_2 \max_{i\in[n]} \|\mathbf{x}_i\|_2^2}{nm}} + \sqrt{\frac{d^2 \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \|\mathbf{C}\|_2}{nm^2}} \\
&\quad + \frac{d \max_{i\in[n]} \|\mathbf{x}_i\|_2^2}{nm}\Big) \\
&\leq \widetilde{O}\Big(\frac{\tau d}{m}\sqrt{\frac{\|\mathbf{C}\|_2}{n}} + \frac{\tau^2 d}{nm}\Big).
\end{aligned} \tag{5.88}$$

In summary,

$$\begin{aligned}
&\|\mathbf{C}_e - \mathbf{C}\|_2 \\
&\leq \min\{\widetilde{O}\Big(\tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2}{m}\sqrt{\frac{d}{n}} + \frac{\tau^2 d}{nm}\Big), \widetilde{O}\Big(\frac{\tau d}{m}\sqrt{\frac{\|\mathbf{C}\|_2}{n}} + \frac{\tau^2 d}{nm}\Big)\}.
\end{aligned} \tag{5.89}$$

For Eq. (5.86) in Theorem 5.8, we first simplify its R and σ^2. According to [10], to obtain a more accurate estimation, each $\mathbf{x}_i$ is required to be multiplied by $\mathbf{HD}$ to flatten its large entries before being sampled uniformly without replacement, where $\mathbf{H}$ is a Hadamard matrix with its dimension being 2^l (l is a certain positive integer), and $\mathbf{D}$ is a diagonal matrix with its diagonal elements being i.i.d. Rademacher random variables. Note that $\mathbf{HDD}^T\mathbf{H}^T = \mathbf{D}^T\mathbf{H}^T\mathbf{HD}$ is an identity matrix.

Suppose that we do not have to pad $\mathbf{X}$ with zeros until its dimension $d = 2^l$ holds. Hence, assuming that $d = 2^l$ for $\mathbf{X} \in \mathbb{R}^{d\times n}$ without loss of generality, we define $\mathbf{Y} = \mathbf{HDX} \in \mathbb{R}^{d\times n}$ below.

Corollary 2 of [10] indicates that with probability at least

$1-\beta$, we have

$$\max_{k\in[d],i\in[n]} |y_{ki}| \leq \sqrt{\frac{1}{d}}\sqrt{2\log(\frac{2nd}{\beta})} \max_{i\in[n]} \|\mathbf{x}_i\|_2 \tag{5.90}$$

and

$$\max_{i\in[n]} \|\mathbf{y}_i\|_2 \leq \sqrt{2\log(\frac{2nd}{\beta})} \max_{i\in[n]} \|\mathbf{x}_i\|_2. \tag{5.91}$$

Corollary 3 of [10] indicates that with probability at least $1-\beta$, we have

$$\|\mathbf{S}_i\mathbf{S}_i^T\mathbf{y}_i\|_2 \leq \sqrt{\frac{m}{d}}\sqrt{2\log(\frac{2nd}{\beta})}\|\mathbf{x}_i\|_2. \tag{5.92}$$

To make a compact representation, we define $\theta = \sqrt{2\log(\frac{2nd}{\beta})}$. Obviously, $\theta > 1$.

Then, in Theorem 5.8, we can replace the input data $\mathbf{X}$ by $\mathbf{Y}$. Combing Eq. (5.92) with the fact that $\|\mathbf{y}_i\|_2 = \|\mathbf{HDx}_i\|_2 = \|\mathbf{x}_i\|_2$ getting $\rho = ((\sqrt{\frac{m}{d}}\theta)^2) = \frac{m\theta^2}{d}$ for the setting of Theorem 5.8. Along with $\theta > 1$ and $m \leq d$, we have

$$\begin{aligned}
& R \\
&= \frac{1}{n}O\Big((\frac{d^2}{m^2}\frac{m\theta^2}{d}+1)\theta^2 \max_{i\in[n]}\|\mathbf{x}_i\|_2^2 + \frac{d(d-m)}{m^2}(\sqrt{\frac{1}{d}}\theta)^2 \max_{i\in[n]}\|\mathbf{x}_i\|_2^2\Big) \\
&= O\Big((\frac{d\theta^2}{nm})\theta^2 \max_{i\in[n]}\|\mathbf{x}_i\|_2^2\Big) \\
&= \widetilde{O}\Big(\frac{d}{nm}\max_{i\in[n]}\|\mathbf{x}_i\|_2^2\Big) \\
&= \widetilde{O}\Big(\frac{\tau^2 d}{nm}\Big),
\end{aligned} \tag{5.93}$$

and

$$\begin{aligned}
&\sigma^2 \\
&\leq \frac{d^2}{nm^2} O\Big((\frac{m\theta^2}{d} - \frac{m(m-1)}{d(d-1)})\theta^2 \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \| \frac{\mathbf{HDXX}^T\mathbf{D}^T\mathbf{H}^T}{n} \|_2 \qquad (5.94) \\
&\quad + \frac{(d-m)}{m}\frac{m\theta^2}{d}\theta^2 \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \|\mathbb{D}(\frac{\mathbf{HDXX}^T\mathbf{D}^T\mathbf{H}^T}{n})\|_2 \\
&\quad + \frac{d-m}{nm}\frac{\theta^2}{d} \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \|\mathbf{HDX}\|_F^2 + \frac{(d-m)^2}{ndm} n \frac{\theta^4}{d^2} \max_{i\in[n]} \|\mathbf{x}_i\|_2^4 \Big) \\
&= \frac{d^2}{nm^2} O\Big(\frac{m}{d}(\theta^2 - \frac{m-1}{d-1})\theta^2 \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \|\mathbf{C}\|_2 \\
&\quad + \frac{(d-m)\theta^4}{d} \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \frac{n(\sqrt{1/d}\theta)^2 \max_{i\in[n]} \|\mathbf{x}_i\|_2^2}{n} \\
&\quad + \frac{(d-m)\theta^2}{nmd} \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 nd \frac{\theta^2}{d} \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \\
&\quad + \frac{(d-m)^2\theta^4}{d^3 m} \max_{i\in[n]} \|\mathbf{x}_i\|_2^4 \Big) \\
&= \widetilde{O}\Big(\frac{d}{nm} \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \|\mathbf{C}\|_2 + \frac{d-m}{nm^2} \max_{i\in[n]} \|\mathbf{x}_i\|_2^4 \\
&\quad + \frac{d(d-m)}{nm^3} \max_{i\in[n]} \|\mathbf{x}_i\|_2^4 + \frac{(d-m)^2}{nm^3 d} \max_{i\in[n]} \|\mathbf{x}_i\|_2^4 \Big) \\
&= \widetilde{O}\Big(\frac{d}{nm} \max_{i\in[n]} \|\mathbf{x}_i\|_2^2 \|\mathbf{C}\|_2 + \frac{d(d-m)}{nm^3} \max_{i\in[n]} \|\mathbf{x}_i\|_2^4 \\
&\quad + \frac{(d-m)^2}{nm^3 d} \max_{i\in[n]} \|\mathbf{x}_i\|_2^4 \Big) \\
&= \widetilde{O}\Big(\frac{\tau^2 d \|\mathbf{C}\|_2}{nm} + \frac{\tau^4 d(d-m)}{nm^3} + \frac{\tau^4 (d-m)^2}{nm^3 d} \Big). \qquad (5.95)
\end{aligned}$$

Note that Eq. (5.95) for simplifying σ^2 in Eq. (5.86) is tighter than the simplification result in the original paper [10] that scales with $\frac{d^2}{nm^2}$. Recalling Eq. (5.86), and replacing its ϵ by

R and σ^2 to get that with probability at least $1-\delta-\beta$, we have

$$\begin{aligned}&\|\mathbf{C}_e - \mathbf{C}\|_2 \\ &\leq \widetilde{O}\Big(\tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2}{m}\sqrt{\frac{d(d-m)}{nm}} + \frac{\tau^2(d-m)}{m}\sqrt{\frac{1}{nmd}} + \frac{\tau^2 d}{nm}\Big).\end{aligned} \tag{5.96}$$

If $m = d$, then

$$\|\mathbf{C}_e - \mathbf{C}\|_2 \leq \widetilde{O}\Big(\tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2 d}{nm}\Big). \tag{5.97}$$

Although pure sampling without replacement makes no estimation error when $m = d$, processing the data by a Hadamard matrix before sampling can result in the error as shown in Eq. (5.97).

If $m < d$ with m being close to d, then $d - m = O(1)$, and thus we have

$$\|\mathbf{C}_e - \mathbf{C}\|_2 \leq \widetilde{O}\Big(\tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2}{m}\sqrt{\frac{d}{nm}} + \frac{\tau^2 d}{nm}\Big). \tag{5.98}$$

If $m \ll d$ or there exists a certain constant $\kappa < 1$ with $m < \kappa d$, then $O(d-m) = O(d)$. In addition to considering that $\frac{1}{nd}\|\mathbf{X}\|_F^2 \leq \|\mathbf{C}\|_2 \leq \max_{i\in[n]}\|\mathbf{x}_i\|_2^2 = \tau^2$, then we have

$$\|\mathbf{C}_e - \mathbf{C}\|_2 \leq \widetilde{O}\Big(\tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}} + \frac{\tau^2 d}{m}\sqrt{\frac{1}{nm}} + \frac{\tau^2 d}{nm}\Big). \tag{5.99}$$

5.8 Details for Computational Complexity

Recall that we have n data samples in the d-dimensional space, and let m be the target compressed dimension. The computational comparisons between our proposed method and the other approaches are presented in Table 5.1, in which *Standard* method means computing $\mathbf{C}$ directly without data compression. We should explain some terms in the table before proceeding.

Storage: storing data and random projection matrices (if any) in the remote sites and the fusion center, and storing the covariance matrix in the fusion center.

Communication: shipping the data and random projection matrices (if any) from remote sites to the fusion center (*high communication cost requires tremendous bandwidth and power consumption*).

Time (FLOPS): compressing the data in the remote sites, and calculating the covariance matrix in the fusion center (*a low time complexity means a low power cost and high efficiency for the data processing*).

Note that, instead of only using the fusion center, data have to be first collected from many remote sites like a network of $g \ll n$ sensors. Then, they are transmitted to the fusion center to estimate the covariance matrix. This procedure shows why communication cost is required. In the table, except for the communication, the two other compared terms have contained the total costs in both the remote sites and fusion center.

For a covariance matrix defined as $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T - \bar{\mathbf{x}}\bar{\mathbf{x}}^T$, we can exactly calculate $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ in the fusion center by $\bar{\mathbf{x}} = \frac{1}{n}\sum_{j=1}^{g}\mathbf{u}_j$, where $\{\mathbf{x}_i\}_{i=1}^{n}$ are distributed in $g \ll n$ remote sites, and $\mathbf{u}_j \in \mathbb{R}^d$ is the summation of all data vectors in the j-th remote site before being compressed. Hence, about $O(gd)$ storage, $O(gd)$ communication cost, and $O(nd)$ time have to be added to the last four methods in Table 5.1, with $g \ll n$.

From now on, we can focus on the covariance matrix defined as $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$.

First, we derive the computational costs in our propose algorithm. Computing $\{p_{ki}\}_{k\in[d],i\in[n]}$ takes $O(nd)$ time. Then, sampling nm entries from all data vectors to get $\mathbf{Y} \in \mathbb{R}^{m\times n}$ takes time that is scaled on $nm\log d$ up to a certain small constant. In Eq. (5.1), each $\mathbf{S}_i$, $\mathbf{S}_i^T\mathbf{x}_i$, $\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i$, and $\mathbf{S}_i\mathbf{S}_i^T$ (squared diagonal), has at most m non-zero entries. Hence, recovering $\{\mathbf{S}_i\}_{i=1}^n$ via the sampled nm entries in $\mathbf{Y}$ and the sampling indices in $\mathbf{T} \in \mathbb{R}^{m\times n}$ incurs $O(nm)$ time. With $\mathbf{Y}$ and $\mathbf{T}$ in hand, $\{\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\}_{i=1}^n$ can be accurately computed in $O(nm)$ time. Equipped with $\{\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\}_{i=1}^n$, computing $\widehat{\mathbf{C}}_1 = \frac{m}{nm-n}\sum_{i=1}^n \mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$ additionally takes only $O(nm^2)$ time, this is due to that each $\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T \in \mathbb{R}^{d\times d}$ has at most m and m^2 non-zero entries respectively. Based on the obtained $\widehat{\mathbf{C}}_1$, computing the square diagonal matrix $\widehat{\mathbf{C}}_2 = \frac{m}{nm-n}\sum_{i=1}^n \mathbb{D}(\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T)\mathbb{D}(\mathbf{b}_i)$ takes $O(nm)$ time since each $\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$ has at most m non-zero entries in its diagonal. Finally, obtaining $\mathbf{C} = \widehat{\mathbf{C}}_1 - \widehat{\mathbf{C}}_2$ incurs $O(d)$ extra time. The total running time is about $O(nd + nm\log d + nm + nm + nm + nm^2 + nm + d) = O(nd + nm\log d + nm^2)$. In the remote sites, data are compressed into m dimensional space. Computing b_{ki} only corresponding to the sampled entries is enough to exactly calculate the $\widehat{\mathbf{C}}_2 = \frac{m}{nm-n}\sum_{i=1}^n \mathbb{D}(\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T)\mathbb{D}(\mathbf{b}_i)$ in Eq. (5.1), so that at most nm entries from $\{p_{ki}\}_{k\in[d],i\in[n]}$ have to be retained to obtain $\{b_{ki}\}$, since $b_{ki} = \frac{1}{1+(m-1)p_{ki}}$. Thus, in the remote sites, $\mathbf{Y} \in \mathbb{R}^{m\times n}$ and $\mathbf{T} \in \mathbb{R}^{m\times n}$ dominate the storage cost, taking about $O(nm)$ space in total. In the fusion center, $O(d^2)$ storage is additionally used to store the estimated covariance $\mathbf{C}_e \in \mathbb{R}^{d\times d}$. Similarly, about $O(nm)$ communication cost is required because of transmitting $\mathbf{Y} \in \mathbb{R}^{m\times n}$, $\mathbf{T} \in \mathbb{R}^{m\times n}$, $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{w} \in \mathbb{R}^n$ and α.

Then, for *Standard* in Table 5.1 that means directly calculating covariance matrix through the observed data samples without compression, it is straightforward to check its computational complexity. $\mathbf{X} \in \mathbb{R}^{d\times n}$ and $\mathbf{C} \in \mathbb{R}^{d\times d}$ takes about $O(nd+d^2)$ storage in total, and $\mathbf{X} \in \mathbb{R}^{d\times n}$ leads to about $O(nd)$ communication burden. Calculating the covariance matrix $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ costs $O(nd^2)$ time.

For *Gauss-Inverse*, $\sum_{i=1}^{n} \mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T$, which is the main part of its unbiased estimator, dominates the computational cost. Generating n different Gaussian matrices $\{\mathbf{S}_i \in \mathbb{R}^{d\times m}\}_{i=1}^{n}$ by the pseudorandom number generator like Mersenne twister [137], which is by far the most widely used, takes considerably large amount of time in practice. The time cost can be denoted by T_G. As $\mathbf{S}_i$ is dense, computing $\{\mathbf{S}_i^T\mathbf{x}_i\}_{i=1}^{n}$ takes $O(nmd)$ time. Calculating $\{(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\}_{i=1}^{n}$ requires $O(nm^2d + nm^3)$, which involves matrix multiplications and inversions. Subsequently, we repeat the matrix-vector multiplications in $\{\mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^{n}$ from the left to right, based on which we get the target covariance matrix. Finally, it takes at least $O(nmd + nm^2d + nm^3 + nm^2 + ndm + nd^2) + T_G = O(nmd + nm^2d + nd^2) + T_G$ time for *Gauss-Inverse*. In the remote sites, we compress data by $\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^m$ before sending them to the fusion center. Along with $O(d^2)$ storage for the derived covariance matrix, about $O(nm+d^2)$ storage space is required in total. Also, sending $\{\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^m\}_{i=1}^{n}$ requires about a $O(nm)$ computational burden.

Note that we have not listed the synchronization cost of *Gauss-Inverse* in Table 5.1. In practice, a pseudo-random number generator is applied to the program in both the remote sites and the fusion center to generate/reconstruct n Gaussian random matrices $\{\mathbf{S}_i \in \mathbb{R}^{d\times m}\}_{i=1}^{n}$, and only n seeds are required to be transmitted from remote sites to the fusion center to recover the Gaussian random matrices. Therefore, only about $O(n)$ storage and communication cost have to be added in Table 5.1. Also, calculating each $(\mathbf{S}_i^T\mathbf{S}_i)^{-1}$ has to load each $\mathbf{S}_i^T\mathbf{S}_i \in \mathbb{R}^{m\times m}$ into memory, hence at least $O(m^2)$ memory is required.

For *Sparse*, calculating $\sum_{i=1}^n \mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$ and subtracting its rescaled diagonal entries dominate the computational cost [9]. Generating sparse projection matrices $\{\mathbf{S}_i \in \mathbb{R}^{d\times q}\}_{i=1}^n$ is also expensive [10], whose time cost is denoted by T_S. The entries of each $\mathbf{S}_i$ are distributed on $\{-1, 0, 1\}$ with probabilities $\{\frac{1}{2s}, 1 - \frac{1}{s}, \frac{1}{2s}\}$. Then, each column of $\mathbf{S}_i$ has $\frac{d}{s}$ non-zero entries in expectation. Empirically, we can fix that $q/d = 0.2$ or 0.4 according to [11, 9]. The number of non-zero entries of $\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^d$ is at least $d(1-(1-\frac{1}{s})^q)$ in expectation, which ranges from $\frac{dq}{s}(1-\frac{q}{2s})$ to $\frac{dq}{s}$. Define $d(1-(1-\frac{1}{s})^q) = m < d$, thus we can solve s with $q/d = 0.2$ or 0.4 fixed to obtain that $s = O(\frac{d^2}{m})$. Then computing $\{\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^q\}_{i=1}^n$ takes $O(\frac{ndq}{s}) = O(nm)$ time in expectation. Based on it, computing $\{\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ additionally costs $O(\frac{ndq}{s}) = O(nm)$ time in expectation. Since each $\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^d$ contains only m non-zeros entries in expectation, thus obtaining $\sum_{i=1}^n \mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$ and subtracting its rescaled diagonal entries requires $O(nm + nm + nm^2 + d) + T_S = O(nm^2 + d) + T_S$ time in total. Storing $\{\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ and the estimated covariance matrix requires $O(nm + d^2)$ storage in expectation, where a $O(nm)$ cost results from $O(nm)$ non-zero entries in $\{\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ along with $O(nm)$ corresponding indices. Similarly, sending $\{\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ from remote sites to the fusion center takes at most $O(nm)$ communication cost in expectation.

For *UniSample-HD*, processing data by a Hadamard matrix by $\mathbf{HDX} \in \mathbb{R}^{d\times n}$ requires $O(nd \log d)$ time, where $\mathbf{H} \in \mathbb{R}^{d\times d}$ can be a Hadamard matrix, $\mathbf{D} \in \mathbb{R}^{d\times d}$ is a diagonal matrix with diagonal elements being i.i.d. Rademacher random variables, and we suppose that $d = 2^l$ holds (l is a certain positive integer). Then, sampling m entries uniformly without replacement on each data vector by $\{\mathbf{S}_i^T\mathbf{HD}\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ takes $O(nm)$ time. Hence, it is straightforward to check that $\sum_{i=1}^n \mathbf{HDS}_i\mathbf{S}_i^T\mathbf{D}^T\mathbf{H}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{D}^T\mathbf{H}^T\mathbf{S}_i\mathbf{S}_i^T\mathbf{HD} \in \mathbb{R}^{d\times d}$ requires $O(nd\log d + nm + nm^2 + d^2\log d) = O(nd\log d + nm^2)$ time in total. $\mathbf{HD} \in \mathbb{R}^{d\times d}$ can be generated on the fly when we process the data. About $O(nm + d^2)$ storage has to be used for the compressed data and estimated covariance matrix. Obviously, about $O(nm)$ communication cost is required.

5.9 Impact of the Parameter α for Our Approach

5.9.1 Discussion

To determine if the k-th entry of the data vector $\mathbf{x}_i \in \mathbb{R}^d$ should be retained or not, the sampling probability applied in our method is

$$p_{ki} = \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}. \tag{5.100}$$

Achieving our theoretical bound of Theorem 5.2 requires $0 < \alpha < 1$. However, The case $\alpha = 1$ and $\alpha = 0$ can also obtain weaker error bounds, which can be straightforwardly derived from Eqs. (5.67)(5.68) and Eqs. (5.73)(5.74). The following illustration reveals the connection between α and error bounds on data owning different properties.

1. Only using $\alpha = 0$, i.e., ℓ_2-norm based sampling $p_{ki} = \frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$ can yield a *very* weak bound if there exist some very small entries $|x_{ki}|$ in $\mathbf{x}_i \in \mathbb{R}^d$. E.g., substituting $p_{ki} = \frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$ into the term $\max_{k\in[d]} \frac{x_{ki}^2}{p_{ki}^2}$ of Eq. (5.67) or Eq. (5.73) results in $\max_{k\in[d]} \frac{\|\mathbf{x}_i\|_2^4}{x_{ki}^2}$ in the final error bound, which becomes infinite if the positive entry $|x_{ki}|$ gets close to 0;

2. Only using $\alpha = 1$, i.e., ℓ_1-norm based sampling $p_{ki} = \frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}$ yields a *slightly* weak bound if there exist some very large entries $|x_{ki}|$ in $\mathbf{x}_i \in \mathbb{R}^d$. E.g., substituting $p_{ki} = \frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}$ into the term $\max_{k\in[d]} \frac{x_{ki}^4}{p_{ki}^2}$ of Eq. (5.73) results in $\max_{k\in[d]} x_{ki}^2\|\mathbf{x}_i\|_1^2$ in the final error bound, which is always greater than or equal to $\max_{k\in[d]} \|\mathbf{x}_i\|_2^4 = \|\mathbf{x}_i\|_2^4$ derived by employing $p_{ki} = \frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$ to bound $\max_{k\in[d]} \frac{x_{ki}^4}{p_{ki}^2}$. Specifically, assume $\|\mathbf{x}_i\|_2^4 = 1$ without loss of generality, then it is possible that $\max_{\mathbf{x}_i\subset\mathbb{R}^d, \|\mathbf{x}_i\|_2^4=1} \max_{k\in[d]} x_{ki}^2\|\mathbf{x}_i\|_1^2 = \frac{d+2\sqrt{d}+1}{4} \gg 1$ if when $x_{ji} = \sqrt{\frac{\sqrt{d}+1}{2\sqrt{d}}}$ and $x_{ki,k\neq j} = \sqrt{\frac{1}{2d+2\sqrt{d}}}$ for all $k \in [d]$ with $k \neq j$. Also, $\min_{\mathbf{x}_i\subset\mathbb{R}^d, \|\mathbf{x}_i\|_2^4=1} \max_{k\in[d]} x_{ki}^2\|\mathbf{x}_i\|_1^2 = 1$ if we have $x_{ki} = \sqrt{\frac{1}{d}}$ for all $k \in [d]$ or we have $x_{ji} = 1$ and $x_{ki,k\neq j} = 0$ for all $k \in [d]$ with $k \neq j$. Note $\mathbf{x}_i \subset \mathbb{R}^d$ in the above optimizations means that $\mathbf{x}_i$ is a vector variable in the d-dimensional space, and j is an arbitrary integer in the set $[d]$.

3. Therefore, α balances the performance by ℓ_1-norm based sampling and ℓ_2-norm based sampling. ℓ_2 sampling penalizes small entries more than ℓ_1 sampling, hence ℓ_2 sampling is more likely to select larger entries to decrease error (e.g., case 2). However, different from ℓ_1 sampling, ℓ_2 sampling is unstable and sensitive to small entries, and it can make

estimation error incredibly high if extremely small entries are picked (e.g., case 1). Then $0 < \alpha < 1$ is applied to achieve the desired tight bound with $p_{ki} \geq (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$ to tackle the extreme situation in the case 2 that cannot be well handled purely by $p_{ki} \geq \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1}$. When α turns from 1 to 0, the estimation error is likely to first decrease and then increase.

5.9.2 Experiments

Accordingly, we create four different synthetic datasets: $\{\mathbf{A}_i\}_{i=1}^4 \in \mathbb{R}^{1000\times 10000}$ (i.e., $d = 1000$ and $n = 10000$). All entries in $\mathbf{A}_1$ and $\mathbf{A}_2$ are i.i.d. generated from the Gaussian distributions $\mathcal{N}(\sqrt{\frac{1}{2d+2\sqrt{d}}}, \frac{1}{1000})$ and $\mathcal{N}(\sqrt{\frac{1}{2d+2\sqrt{d}}}, \frac{1}{100})$, respectively. For $\mathbf{A}_3$, the entries of its one row are i.i.d. generated from $\mathcal{N}(\sqrt{\frac{\sqrt{d}+1}{2\sqrt{d}}}, \frac{1}{100})$, and the other entries follow $\mathcal{N}(\sqrt{\frac{1}{2d+2\sqrt{d}}}, \frac{1}{100})$. For $\mathbf{A}_4$, its generation follows the way of $\mathbf{X}_1$ in Section 5.4.

In Figure 5.5, the y-axis reports the errors that are normalized by the error incurred at $\alpha = 1$. For $\mathbf{A}_1$, the magnitudes of the data entries tend to be *highly* uniformly distributed. Thus, nearly the same results are returned over all α. For $\mathbf{A}_2$, its entries are *slightly* uniformly distributed with some entries having extremely small magnitudes. Hence, $\alpha = 0$ has a poorer performance compared with the others, which is consistent with the case 1 in Section 5.9.1. $\mathbf{A}_3$ contains some entries larger than the others, and neither $\alpha = 0$ nor $\alpha = 1$ achieves the best performance obtained roughly at $\alpha = 0.9$. Also, the estimation error first decreases and then increases when α turns from 1 to 0. All such simulation results conform to the case 2 and case 3 in Section 5.9.1. Considering $\mathbf{A}_4$ that is not likely to contain the extreme situation as mentioned in the case 2 of Section 5.9.1, we see that best performance is roughly achieved when α gets close to 1.

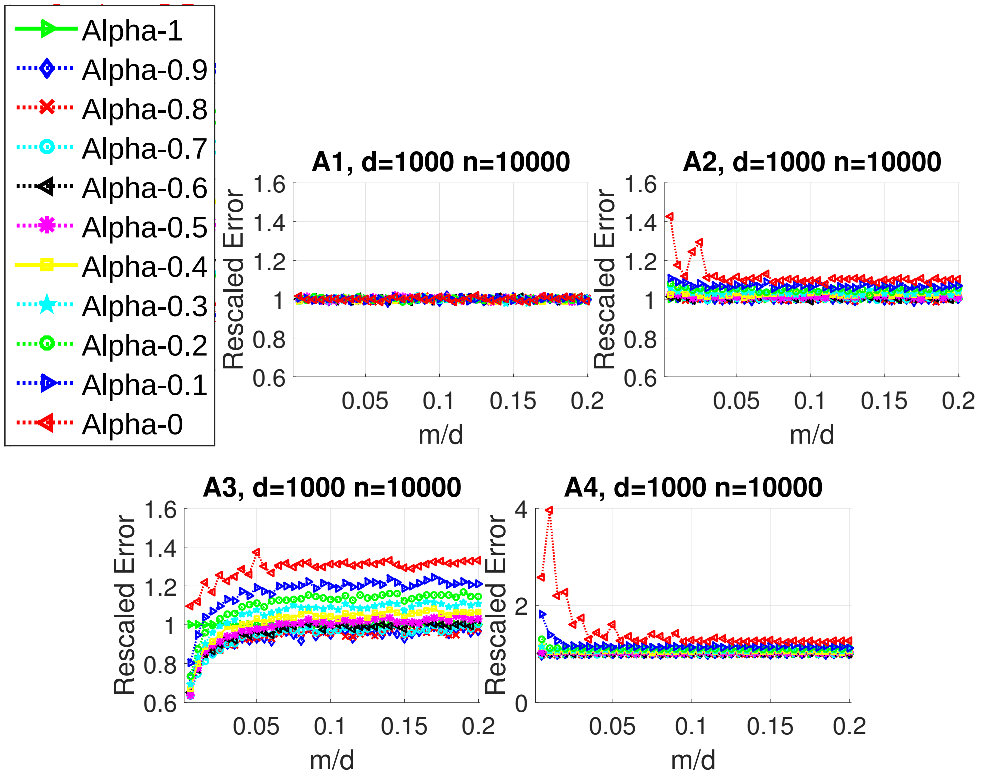

Figure 5.5: Accuracy comparison by decreasing α from 1 to 0 with a step size of 0.1. The error at each α is normalized by that at $\alpha = 1$ on y-axis, and m/d varies from 0.005 to 0.2 with a step size of 0.005 on x-axis. Roughly, $\alpha = 0.9$ is a good choice, and the smaller parameter like $\alpha = 0$ usually leads to a poorer accuracy and higher variance compared with the other α values.

□ **End of chapter.**

Chapter 6

Conclusions and Future Work

In this chapter, we summarize the main contribution of this thesis and provide several interesting future directions.

6.1 Conclusion

In this thesis, we focus on the line of research on exploiting randomized algorithms to enable learning on big data. For the learning problems including kernel methods, unsupervised online hashing, and covariance matrix estimation, we have developed practical randomized algorithms with superior theoretical guarantees. We have also demonstrated the effectiveness through extensive experiments.

In Chapter 3, we leverage random projections including Gaussian projection matrix and fast JL transform to make the training of kernel methods efficient. Instead of directly applying random projection matrices to refine the mapped kernel features, we efficiently find the dominant low-dimensional space of kernel features via projection matrices and then project the mapped kernel features into such a low-dimensional space. Through this way, we can significantly reduce the volume of kernel features, and make the training easy.

In Chapter 4, we employ distinct fast JL transforms to accelerate the state-of-the-art unsupervised online hashing algorithm. Fast JL transform can compress data efficiently, and we propose to utilize distinct transforms to guarantee the compression accuracy. Moreover, employing fast JL transform for streaming data requires more space, and we derive a novel implementation of fast JL transform so that the space efficiency can still be maintained. The computational advantages of our method are supported by both the theoretical analysis and empirical results.

Finally, in Chapter 5, we develop weighted random sampling with carefully designed probability distributions for covariance matrix estimation. Compared with the other three randomized algorithms, our method achieves both the best estimation error bound and computational complexity. Experiments also agree well with our theoretical findings and demonstrate the superiority.

6.2 Future Work

We have investigated the multiple applications of the randomized algorithm paradigm to make the learning efficient. There are some other interesting directions left to be explored in the future.

6.2.1 Randomized Algorithms and Implicit Regularization

In the theoretical analysis, our thesis and many other studies only quantify that the outputs of the randomized algorithms sacrifice a little accuracy on the observed data in comparison with the corresponding deterministic algorithms. However, in practice, we care more about the learning accuracy on the unobserved data (i.e., testing data). In addition, in some cases, the learning accuracy of randomized algorithms on the unobserved data can even outperform the deterministic algorithms, which, for example, can also be observed from the empirical results of the last dataset in Figure 4.4 of Chapter 4. This superiority has actually been theoretically justified in the randomized algorithms for high-dimensional least regressions [66, 135], and the derived theoretical results therein on the unobserved data imply that randomized algorithms play a role of implicit regularization in the generalization ability. Therefore, it is also important to analyze the learning accuracy of the randomized algorithms studied in this thesis for the unobserved data, which in return guides us to develop more appropriate randomized algorithms.

6.2.2 Randomized Algorithms for Deep Neural Networks

Deep neural networks have been the state-of-the-art techniques in many fields such as computer vision and natural language processing. However, deep neural networks are both computationally expensive and space intensive, making it difficult to train and infer on them and to deploy them with limited hardware resources. LeCun et al. have studied dropping unimportant weights in the networks [43]. Recent work [52] shows that there is a large amount of redundancy in the weights of the deep networks by training the low-rank approximations of the weight matrix. Similarly, the work in [42] trains the deep networks with a low bit precision. Such evidences demonstrate the information redundancy in the network. Motivated by these findings, [36, 83] have leveraged randomized algorithms to make a compact representation for certain neural networks while guaranteeing the learning ability, so that the computational burdens are greatly reduced. Thus, it is significant to develop randomized algorithms for a better tradeoff and for more variations of neural networks (e.g., recurrent neural network, deep graph network, etc. [95]).

6.2.3 Randomized Algorithms for Parallel/Distributed Computation

Parallel/distributed computation is currently a popular strategy to enable learning with big data. However, the communication cost among multiple cores in a single machine or across distributed machines is expensive. Considering that computations for randomization are easily parallelized and distributed, randomized algorithms can be leveraged to reduce the communication costs in parallel/distributed computation while maintaining the communicated information well. Chapter 5 and a recent paper by [105] also demonstrate the effectiveness with certain distributed cases. In the future, we plan to develop randomized algorithms for more distributed learning problems and to make the best of modern computer architecture and system requirements [130].

□ **End of chapter.**

Bibliography

[1] S. Abbasi-Daresari and J. Abouei. Toward cluster-based weighted compressive data aggregation in wireless sensor networks. *Ad Hoc Networks*, 36:368–385, 2016.

[2] R. Abrahamsson, Y. Selen, and P. Stoica. Enhanced covariance matrix estimators in adaptive beamforming. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–969. IEEE, 2007.

[3] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[4] D. Achlioptas, Z. S. Karnin, and E. Liberty. Near-optimal entrywise sampling for data matrices. In *Advances in Neural Information Processing Systems*, pages 1565–1573, 2013.

[5] D. Achlioptas and F. Mcsherry. Fast computation of low-rank matrix approximations. *Proceedings of the annual ACM symposium on Theory of computing*, 54(2):9, 2007.

[6] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Pro-*

ceedings of the thirty-eighth annual ACM symposium on Theory of computing, pages 557–563. ACM, 2006.

[7] N. Ailon and E. Liberty. Fast dimension reduction using rademacher series on dual bch codes. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, 2008.

[8] L. Amsaleg. Datasets for approximate nearest neighbor search, 2010.

[9] F. Anaraki. Estimation of the sample covariance matrix from compressive measurements. *IET Signal Processing*, 2016.

[10] F. Anaraki and S. Becker. Preconditioned data sparsification for big data with applications to pca and k-means. *arXiv preprint arXiv:1511.00152*, 2015.

[11] F. P. 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-14)*, pages 1341–1349, 2014.

[12] T. Anderson. *The theory and practice of online learning.* Athabasca University Press, 2008.

[13] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, 2015.

[14] Y. Anzai. *Pattern Recognition & Machine Learning.* Elsevier, 2012.

[15] H. Avron, H. Nguyen, and D. Woodruff. Subspace embeddings for the polynomial kernel. In *Advances in Neural Information Processing Systems*, 2014.

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

[17] D. Bartz. Advances in high-dimensional covariance matrix estimation. 2016.

[18] C. Battaglino, G. Ballard, and T. G. Kolda. A practical randomized cp tensor decomposition. *arXiv preprint arXiv:1701.06600*, 2017.

[19] R. Bekkerman, M. Bilenko, and J. Langford. *Scaling up machine learning: Parallel and distributed approaches.* Cambridge University Press, 2011.

[20] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[21] J. M. Bioucas-Dias, D. Cohen, and Y. C. Eldar. Covalsa: Covariance estimation from compressive measurements using alternating minimization. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pages 999–1003. IEEE, 2014.

[22] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[23] C. Blake and C. Merz. UCI repository of machine learning databases, 1998. URL: http://archive.ics.uci.edu/ml/.

[24] C. Boutsidis, P. Drineas, and M. W. Mahoney. Unsupervised feature selection for the k-means clustering problem. In *Advances in Neural Information Processing Systems*, pages 153–161, 2009.

[25] C. Boutsidis and D. P. Woodruff. Optimal cur matrix decompositions. *SIAM Journal on Computing*, 46(2):543–589, 2017.

[26] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas. Randomized dimensionality reduction for k-means clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, 2015.

[27] A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, and I. S. Kohane. Discovering functional relationships between rna expression and chemotherapeutic susceptibility using relevance networks. *Proceedings of the National Academy of Sciences*, 97(22):12182–12186, 2000.

[28] T. T. Cai, A. Zhang, et al. Rop: Matrix recovery via rank-one projections. *The Annals of Statistics*, 43(1):102–138, 2015.

[29] F. Cakir and S. Sclaroff. Adaptive hashing for fast similarity search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1044–1052, 2015.

[30] F. Cakir and S. Sclaroff. Online supervised hashing. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2606–2610. IEEE, 2015.

[31] D. Calandriello, A. Lazaric, and M. Valko. Second-order kernel online convex optimization with adaptive sketching. In *International Conference on Machine Learning*, 2017.

[32] N. Cesa-Bianchi, Y. Mansour, and O. Shamir. On the complexity of learning with kernels. *arXiv preprint arXiv:1411.1158*, 2014.

[33] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[34] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[35] S. Chen, Y. Liu, M. R. Lyu, I. King, and S. Zhang. Fast relative-error approximation algorithm for ridge regression. In *UAI*, pages 201–210, 2015.

[36] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.

[37] X. Chen, M. R. Lyu, and I. King. Toward efficient and accurate covariance matrix estimation on compressed data. In *International Conference on Machine Learning*, pages 767–776, 2017.

[38] Y. Chen, Y. Chi, and A. Goldsmith. Exact and stable covariance estimation from quadratic sampling via convex programming. 2013.

[39] D. Cheng, R. Peng, Y. Liu, and I. Perros. Spals: Fast alternating least squares via implicit leverage scores sampling. In *Advances In Neural Information Processing Systems*, pages 721–729, 2016.

[40] A. Choromanska, K. Choromanski, M. Bojarski, T. Jebara, S. Kumar, and Y. LeCun. Binary embeddings with structured hashed projections. In *ICML*, 2016.

[41] C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In *International Conference on Artificial Intelligence and Statistics*, pages 113–120, 2010.

[42] M. Courbariaux, J.-P. David, and Y. Bengio. Low precision storage for deep learning. *arXiv preprint arXiv:1412.7024*, 2014.

[43] Y. L. Cun, J. S. Denker, and S. A. Solla. Optimal brain damage, 1990.

[44] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014.

[45] G. Dasarathy, P. Shah, B. N. Bhaskar, and R. D. Nowak. Sketching sparse matrices, covariances, and graphs via tensor products. *Information Theory, IEEE Transactions on*, 61(3):1373–1388, 2015.

[46] S. Dasgupta. Experiments with random projection. In *Proceedings of the Sixteenth conference on Uncertainty in*

artificial intelligence, pages 143–151. Morgan Kaufmann Publishers Inc., 2000.

[47] S. Dasgupta and A. Gupta. An elementary proof of the johnson-lindenstrauss lemma. *International Computer Science Institute, Technical Report*, pages 99–006, 1999.

[48] T. H. Davenport, P. Barth, and R. Bean. How big data is different. *MIT Sloan Management Review*, 54(1):43, 2012.

[49] C. Davis and W. M. Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

[50] V. Dehdari and C. V. Deutsch. Applications of randomized methods for decomposing and simulating from large covariance matrices. In *Geostatistics Oslo 2012*, pages 15–26. Springer, 2012.

[51] M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[52] M. Denil, B. Shakibi, L. Dinh, N. de Freitas, et al. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156, 2013.

[53] P. Dhillon, Y. Lu, D. P. Foster, and L. Ungar. New subsampling algorithms for fast least squares regression. In *Advances in Neural Information Processing Systems*, pages 360–368, 2013.

[54] E. Drinea, P. Drineas, and P. Huggins. A randomized singular value decomposition algorithm for image processing applications. In *In Proceedings of the 8th panhellenic conference on informatics*, pages 278–288, 2001.

[55] P. Drineas and R. Kannan. Pass efficient algorithms for approximating large matrices. In *SODA*, volume 3, pages 223–232, 2003.

[56] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[57] P. Drineas and M. W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(Dec):2153–2175, 2005.

[58] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization*. 2006.

[59] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

[60] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2011.

[61] R. J. Durrant and A. Kaban. Compressed fisher linear discriminant analysis: Classification of randomly pro-

jected data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1119–1128. ACM, 2010.

[62] R. J. Durrant and A. Kaban. Random projections as regularizers: learning a linear discriminant from fewer observations than dimensions. *Machine Learning*, 99(2):257–286, 2015.

[63] A. El Alaoui and M. W. Mahoney. Fast randomized kernel methods with statistical guarantees. *stat*, 1050:2, 2014.

[64] P. C. Evans and M. Annunziata. Industrial internet: Pushing the boundaries. *General Electric Reports*, 2012.

[65] J. Fan, F. Han, and H. Liu. Challenges of big data analysis. *National science review*, 1(2):293–314, 2014.

[66] M. M. Fard, Y. Grinberg, J. Pineau, and D. Precup. Compressed least-squares regression on sparse spaces. In *Association for the Advancement of Artificial Intelligence*, 2012.

[67] W. Feller. introduction to probability theory and its applications. vol. ii.[an]. 1966.

[68] Ó. Fontenla-Romero, B. Guijarro-Berdiñas, D. Martinez-Rego, B. Pérez-Sánchez, and D. Peteiro-Barral. Online machine learning. *Efficiency and Scalability Methods for Computational Intellect*, 27, 2013.

[69] M. H.-Y. Fritz, R. Leinonen, G. Cochrane, and E. Birney. Efficient storage of high throughput dna sequencing data using reference-based compression. *Genome research*, 21(5):734–740, 2011.

[70] M. Ghashami, E. Liberty, and J. M. Phillips. Efficient frequent directions algorithm for sparse matrices. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 845–854. ACM, 2016.

[71] M. Ghashami and J. M. 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. Society for Industrial and Applied Mathematics, 2014.

[72] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[73] A. Gittens. The spectral norm error of the naive nystrom extension. *arXiv preprint arXiv:1110.5305*, 2011.

[74] S. Gleichman and Y. C. Eldar. Blind compressed sensing. *Information Theory, IEEE Transactions on*, 57(10):6958–6975, 2011.

[75] G. H. Golub and C. Van Loan. Matrix computations. 1996.

[76] A. Gonen, F. Orabona, and S. Shalev-Shwartz. Solving ridge regression using sketched preconditioned svrg. In *International Conference on Machine Learning*, pages 1397–1405, 2016.

[77] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[78] W. Ha and R. F. Barber. Robust pca with compressed data. In *Advances in Neural Information Processing Systems*, pages 1936–1944, 2015.

[79] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[80] K. B. Hall, S. Gilpin, and G. Mann. Mapreduce/bigtable for distributed optimization. In *NIPS LCCC Workshop*, 2010.

[81] R. Hamid, Y. Xiao, A. Gittens, and D. DeCoste. Compact random feature maps. 2014.

[82] W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[83] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[84] L. P. Hansen. Large sample properties of generalized method of moments estimators. *Econometrica: Journal of the Econometric Society*, pages 1029–1054, 1982.

[85] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[86] J. Haupt, W. U. Bajwa, M. Rabbat, and R. Nowak. Compressed sensing for networked data. *IEEE Signal Processing Magazine*, 25(2):92–101, 2008.

[87] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[88] C. Heinze, B. McWilliams, and N. Meinshausen. Dual-loco: Distributing statistical estimation using random projections. In *Artificial Intelligence and Statistics*, pages 875–883, 2016.

[89] J. T. Holodnak and I. C. Ipsen. Randomized approximation of the gram matrix: Exact computation and probabilistic bounds. *SIAM Journal on Matrix Analysis and Applications*, 36(1):110–137, 2015.

[90] J. Hromkovic. *Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics.* Springer Science & Business Media, 2013.

[91] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. 2003.

[92] L.-K. Huang, Q. Yang, and W.-S. Zheng. Online hashing. In *IJCAI*. Citeseer, 2013.

[93] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[94] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998.

[95] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

[96] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 2011.

[97] Q.-Y. Jiang and W.-J. Li. Scalable graph hashing with feature transformation. In *IJCAI*, pages 2248–2254, 2015.

[98] R. Jin. Deep learning at alibaba. 2017.

[99] Z. John Lu. The elements of statistical learning: data mining, inference, and prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173(3):693–694, 2010.

[100] W. B. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986.

[101] I. Jolliffe. *Principal component analysis.* Wiley Online Library, 2002.

[102] W.-C. Kang, W.-J. Li, and Z.-H. Zhou. Column sampling based discrete supervised hashing. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[103] T. Kariya and H. Kurata. *Generalized least squares*. John Wiley & Sons, 2004.

[104] R. M. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, 34(1-3):165–201, 1991.

[105] J. Konevcny and P. Richtarik. Randomized distributed mean estimation: Accuracy vs communication. *arXiv preprint arXiv:1611.07555*, 2016.

[106] W. Kong and W.-J. Li. Isotropic hashing. In *Advances in Neural Information Processing Systems*, 2012.

[107] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[108] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[109] Q. Le, T. Sarlós, and A. Smola. Fastfood approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.

[110] M. Ledoux. *The concentration of measure phenomenon*. Number 89. American Mathematical Soc., 2005.

[111] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu. Online sketching hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2503–2511, 2015.

[112] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and

B.-Y. Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 1, page 3, 2014.

[113] P. Li, T. J. Hastie, and K. W. Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006.

[114] W. Li. Learning to hash for big data. *Tutorial*, 2012.

[115] W. Li, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012.

[116] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, 2016.

[117] Y. Li, R. Wang, H. Liu, H. Jiang, S. Shan, and X. Chen. Two birds, one stone: Jointly learning binary code for large-scale face image retrieval and attributes prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[118] E. Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013.

[119] E. Liberty and S. W. Zucker. The mailman algorithm: A note on matrix–vector multiplication. *Information Processing Letters*, 109(3):179–182, 2009.

[120] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin. Large-scale logistic regression and linear support vector machines using spark. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 519–528. IEEE, 2014.

[121] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[122] H. Liu and W.-S. Chen. A novel random projection model for linear discriminant analysis based face recognition. In *Wavelet Analysis and Pattern Recognition, 2009. ICWAPR 2009. International Conference on*, pages 112–117. IEEE, 2009.

[123] H. Liu, R. Ji, Y. Wu, and F. Huang. Ordinal constrained binary code learning for nearest neighbor search. In *AAAI*, 2017.

[124] H. Liu, R. Ji, Y. Wu, and W. Liu. Towards optimal binary code learning via ordinal embedding. In *AAAI*, 2016.

[125] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, pages 3419–3427, 2014.

[126] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proceedings of the 28th international conference on machine learning*, 2011.

[127] D. Lopez-paz, S. Sra, A. Smola, Z. Ghahramani, and B. Schoelkopf. Randomized nonlinear component analysis.

In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1359–1367, 2014.

[128] Y. Lu, P. Dhillon, D. P. Foster, and L. Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems*, 2013.

[129] H. Luo, A. Agarwal, N. Cesa-Bianchi, and J. Langford. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems*, 2016.

[130] M. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[131] M. W. Mahoney. Lecture notes on randomized linear algebra. *arXiv preprint arXiv:1608.04481*, 2016.

[132] M. W. Mahoney and P. Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[133] M. W. Mahoney, M. Maggioni, and P. Drineas. Tensor-cur decompositions for tensor-based data. *SIAM Journal on Matrix Analysis and Applications*, 30(3):957–987, 2008.

[134] O.-A. Maillard and R. Munos. Linear regression with random projections. *Journal of Machine Learning Research*, 13(Sep):2735–2772, 2012.

[135] O.-A. Maillard, R. Munos, et al. Compressed least-squares regression. In *In Advances in Neural Information Processing Systems*, pages 1213–1221, 2009.

[136] P.-G. Martinsson, A. Szlam, M. Tygert, et al. Normalized power iterations for the computation of svd. *Manuscript., Nov*, 2010.

[137] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 1998.

[138] Y. Mroueh, E. Marcheret, and V. Goel. Co-occuring directions sketching for approximate matrix multiply. *arXiv preprint arXiv:1610.07686*, 2016.

[139] L. Mukherjee, S. N. Ravi, V. K. Ithapu, T. Holmes, and V. Singh. An nmf perspective on binary hashing. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[140] C. Musco and C. Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, pages 1396–1404, 2015.

[141] C. Musco and C. Musco. Recursive sampling for the nystrom method. In *Advances in Neural Information Processing Systems*, pages 3836–3848, 2017.

[142] J. Nelson and H. L. Nguyen. Lower bounds for oblivious subspace embeddings. In *Automata, Languages, and Programming*, pages 883–894. Springer, 2014.

[143] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[144] B. Neyshabur, N. Srebro, R. R. Salakhutdinov, Y. Makarychev, and P. Yadollahpour. The power of asymmetry in binary hashing. In *Advances in Neural Information Processing Systems*, 2013.

[145] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pages 442–450, 2015.

[146] D. Papailiopoulos, A. Kyrillidis, and C. Boutsidis. Provable deterministic leverage score sampling. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 997–1006. ACM, 2014.

[147] S. Paul, C. Boutsidis, M. Magdon-Ismail, and P. Drineas. Random projections for support vector machines. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pages 498–506, 2013.

[148] J. Pennington, F. Yu, and S. Kumar. Spherical random features for polynomial kernels. In *Advances in Neural Information Processing Systems*, pages 1846–1854, 2015.

[149] D. Peteiro-Barral and B. Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[150] M. Pilanci. *Fast Randomized Algorithms for Convex Optimization and Statistical Estimation.* PhD thesis, University of California, Berkeley, 2016.

[151] E. Price. Lecture 1 of Randomized Algorithms.

`https://www.cs.utexas.edu/~ecprice/courses/randomized/notes/lec1.pdf`, 2017.

[152] H. Qi and S. M. Hughes. Invariance of principal components under low-dimensional random projection of the data. In *Image Processing (ICIP)*. IEEE, 2012.

[153] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.

[154] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[155] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

[156] W. Rudin. *Fourier analysis on groups*. John Wiley & Sons, 1990.

[157] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann, 1998.

[158] J. Schäfer and K. Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005.

[159] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Computational learning theory*, pages 416–426. Springer, 2001.

[160] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2001.

[161] S. Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[162] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. Tao Shen. Learning binary codes for maximum inner product search. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[163] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[164] T. Shi, D. Tang, L. Xu, and T. Moscibroda. Correlated compressive sensing for networked data. In *UAI*, pages 722–731, 2014.

[165] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.

[166] J. Song, L. Gao, Y. Yan, D. Zhang, and N. Sebe. Supervised hashing with pseudo labels for scalable multimedia retrieval. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 827–830. ACM, 2015.

[167] Z. Song, D. Woodruff, and H. Zhang. Sublinear time orthogonal tensor decomposition. In *Advances in Neural Information Processing Systems*, pages 793–801, 2016.

[168] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki. Practical data compression in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, 35(1):37–59, 2012.

[169] A. T. Suresh, F. X. Yu, H. B. McMahan, and S. Kumar. Distributed mean estimation with limited communication. *arXiv preprint arXiv:1611.00429*, 2016.

[170] J. A. Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011.

[171] J. A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 2015.

[172] A. M. Tulino and S. Verdú. *Random matrix theory and wireless communications*, volume 1. Now Publishers Inc, 2004.

[173] S. R. Upadhyaya. Parallel approaches to machine learning—a comprehensive survey. *Journal of Parallel and Distributed Computing*, 73(3):284–292, 2013.

[174] J. Wang, S. C. Hoi, P. Zhao, J. Zhuang, Z.-Y. Liu, et al. Large scale online kernel classification. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1750–1756. AAAI Press, 2013.

[175] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition, 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010.

[176] J. Wang, J. D. Lee, M. Mahdavi, M. Kolar, N. Srebro, et al. Sketching meets random projection in the dual: A provable recovery algorithm for big and high-dimensional data. *Electronic Journal of Statistics*, 11(2):4896–4944, 2017.

[177] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *arXiv preprint arXiv:1606.00185*, 2016.

[178] P.-A. Wang and C.-J. Lu. Tensor decomposition via simultaneous power iteration. In *International Conference on Machine Learning*, pages 3665–3673, 2017.

[179] S. Wang, A. Gittens, and M. W. Mahoney. Scalable kernel k-means clustering with nystrom approximation: Relative-error bounds. *arXiv preprint arXiv:1706.02803*, 2017.

[180] S. Wang, A. Gittens, and M. W. Mahoney. Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging. *arXiv preprint arXiv:1702.04837*, 2017.

[181] S. Wang and Z. Zhang. Improving cur matrix decomposition and the nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1):2729–2769, 2013.

[182] Y. Wang and A. Anandkumar. Online and differentially-private tensor decomposition. In *Advances in Neural Information Processing Systems*, pages 3531–3539, 2016.

[183] Y. Wang, D. Feng, D. Li, X. Chen, Y. Zhao, and X. Niu. A mobile recommendation system based on logistic re-

gression and gradient boosting decision trees. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 1896–1902. IEEE, 2016.

[184] Y. Wang, H.-Y. Tung, A. J. Smola, and A. Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*, pages 991–999, 2015.

[185] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, 2009.

[186] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *arXiv preprint arXiv:1705.07878*, 2017.

[187] C. K. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[188] D. P. Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 2014.

[189] C. Woolam, M. M. Masud, and L. Khan. Lacking labels in the stream: classifying evolving stream data with few labels. In *International Symposium on Methodologies for Intelligent Systems*, pages 552–562. Springer, 2009.

[190] S. Wu, S. Bhojanapalli, S. Sanghavi, and A. G. Dimakis. Single pass pca of matrix products. In *Advances in Neural Information Processing Systems*, 2016.

[191] L. Xie, L. Zhu, and G. Chen. Unsupervised multi-graph cross-modal hashing for large-scale multimedia retrieval. *Multimedia Tools and Applications*, pages 1–20, 2016.

[192] P. Xu, J. Yang, F. Roosta-Khorasani, C. Ré, and M. W. Mahoney. Sub-sampled newton methods with non-uniform sampling. In *Advances in Neural Information Processing Systems*, pages 3000–3008, 2016.

[193] H. Yang, I. King, and M. R. Lyu. *Sparse Learning Under Regularization Framework.* LAP Lambert Academic Publishing, April 2011.

[194] H. Yang, Z. Xu, J. Ye, I. King, and M. R. Lyu. Efficient sparse generalized multiple kernel learning. *IEEE Transactions on Neural Networks*, 22(3):433–446, March 2011.

[195] J. Yang, Y.-L. Chow, C. Re, and M. W. Mahoney. Weighted sgd for lp regression with randomized preconditioning. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 558–569. Society for Industrial and Applied Mathematics, 2016.

[196] J. Yang, X. Meng, and M. W. Mahoney. Implementing randomized matrix algorithms in parallel and distributed environments. *Proceedings of the IEEE*, 104(1):58–92, 2016.

[197] J. Yang, V. Sindhwani, H. Avron, and M. Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. In *Proceedings of The 31st International Conference on Machine Learning*, pages 485–493, 2014.

[198] T. Yang, Q. Lin, and R. Jin. Big data analytics: Optimization and randomization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2327–2327. ACM, 2015.

[199] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *ICML*, 2014.

[200] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nystrom low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239. ACM, 2008.

[201] L. Zhang, M. Mahdavi, R. Jin, T. Yang, and S. Zhu. Recovering the optimal solution by dual random projection. In *Conference on Learning Theory*, pages 135–157, 2013.

[202] L. Zhang, T. Yang, R. Jin, and Z.-H. Zhou. Sparse learning for large-scale and high-dimensional data: A randomized convex-concave optimization approach. In *International Conference on Algorithmic Learning Theory*, pages 83–97. Springer, 2016.

[203] W. Zhang, L. Zhang, R. Jin, D. Cai, and X. He. Accelerated sparse linear regression via random projection. In *AAAI*, pages 2337–2343, 2016.

[204] Z. Zhang. The singular value decomposition, applications and beyond. *arXiv preprint arXiv:1510.08532*, 2015.

[205] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.