On Efficient Sketching Algorithms

Praneeth Kacham

CMU-CS-24-131

May 2024

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science.*

This research was sponsored by: the Simons Foundation under award number 68963; the National Science Foundation under award number CCF2335412; Indiana University under award number 8029CMUPO0173652; and the Office of Naval Research under award number N000141812562. A part of the work in this thesis was done while the author was a student researcher at Google Research.

Keywords: Sketching, Streaming, Distributed Algorithms, Regression, Low Rank Approximation

I dedicate this dissertation to my parents.

Abstract

Sketching refers to a wide variety of techniques to compress large datasets into much smaller forms that can be *efficiently* processed to answer questions about the original dataset. Over the past few decades, sketching has emerged as a key tool to efficiently handle large datasets in majorly three settings: (i) the *Classic* setting, in which the dataset is given to us and we want to solve a problem as quickly as possible, (ii) the *Streaming* setting, in which the underlying dataset is defined by a large stream of updates and we want to compute interesting properties of the dataset using a small amount of space, and (iii) the *Distributed* setting, in which the dataset of interest is split among multiple servers and we want protocols that use a small amount of communication among servers to solve problems of interest on the underlying dataset.

Each of the above settings presents a different challenge with regard to the measure of efficiency we are interested in. In this thesis, we study sketching algorithms in these three settings for a variety of problems. While the techniques required to obtain our algorithms differ across problems and settings, the underlying idea of (possibly randomized) data compression to convert the original large dataset into a much smaller form is a key ingredient behind all the results in this thesis.

Acknowledgments

First and foremost, I'd like to thank David. I learnt a lot from David about how to approach problems, reassess when stuck, and properly understand the current state of things while doing research. David has been extremely helpful throughout, always willing to spend hours talking about research and introduce new problems and collaborators to me.

I would like to thank Vahab and Peilin for hosting me at Google Research for the last couple of years. I had a lot of fun exploring new research areas beyond the work I did at CMU. I really appreciate Pravesh Kothari, Rasmus Pagh and Richard Peng for agreeing to be on my thesis committee. I am very thankful to Sorav Bansal, Amit Deshpande and Naveen Garg for setting me up on the research path during my undergrad.

I am grateful to my collaborators Ainesh Bakshi, Vincent Cohen-Addad, Kenneth Clarkson, Nadiia Chepurko, Hossein Esfandiari, Zhili Feng, Chirag Gupta, Vahab Mirrokni, Rasmus Pagh, Sandeep Silwal, Mikkel Thorup, Peilin Zhong and Samson Zhou for all the help and the things I've learnt from them.

I fondly cherish all the great moments with my friends and roommates Divyansh, Jatin, Saket and Vishnu over the years in Pittsburgh. It was amazing to share the home with them. I'd like to thank my friends Avaneep, Dinesh, Harshaan, Lovish and Ribhav for keeping me sane at times and making me go insane the other times throughout the PhD years.

I'm thankful to my amazing office mates Asher, Lucio and Suhas for always keeping a fun atmosphere in 5101. I'd also like to thank all the theory group members at CMU for all the talks and chats over the years and inspiring me to work on interesting things. Shout out to Jeff for all the tennis we played! I'd also like to thank Deb Cavlovich, Christina Contreras, Catherine Copetas, Matthew Stewart, Jenn Landefeld, Amanda Hornick and all other CSD staff for always trying to make the life of students easy.

Finally, I'd like to thank my parents Latha and Shekhar and my sister Meghana for always supporting me in whatever I wanted to do.

Contents

Chapter 1

Introduction

In this era of very large datasets, extracting useful insights efficiently has become challenging. Existing algorithms, many of which assume that data can be arbitrarily accessed and that run in super-linear time, have become untenable with growing input sizes. In addition to growing input sizes, another key challenge is the way input to a problem is defined. For example, the input may be arbitrarily partitioned across multiple servers, and we may want to solve a problem on the underlying input with algorithms that are communication efficient. As another example, the input to a problem may be defined by a stream of items, and we may want to solve a problem under the constraint that there is not enough space to store all the items.

Over the past several decades, *sketching* has emerged as an effective paradigm to obtain efficient algorithms to process large datasets in the above described settings. At a high level, sketching refers to the process of efficiently compressing a dataset into a smaller form, called a *sketch*, that contains enough information sufficient to (approximately) solve a problem on the original dataset. Converting datasets into sketches has three major advantages:

1. Due to their small size, time-intensive algorithms such as the Singular Value Decomposition (SVD) can be run much quicker than they can be on the original dataset.
2. If the sketches are efficiently *updateable*, we can obtain small-space algorithms to process data streams by updating the sketch each time the underlying dataset receives an update.
3. Since the sketch can be represented using smaller number of bits than the original dataset, it can be efficiently communicated to other parties who may want to learn properties of the original dataset.

These advantages of sketching have led to efficient algorithms for handling very large datasets. We can broadly classify the large scale data problems into three settings: (i) Classic, (ii) Streaming and (iii) Distributed.

The Classic setting. In this setting, we are given a large $n \times d$ input matrix A, and we want algorithms that run as fast as possible and solve a problem of interest. We preferably want algorithms that run in time $O(\mathrm{nnz}(A))$, where $\mathrm{nnz}(A)$ denotes the number of nonzero entries in A. Such algorithms are usually called *input-sparsity time algorithms.* These algorithms let us support datasets with very large values of n and d assuming that the inputs are sparse. Early literature in numerical linear algebra dealt with obtaining fast and numerically stable algorithms which resulted in very good deterministic algorithms for a number of linear algebra routines such as QR decomposition, Singular Value Decomposition and least-squares regression (see [TB97] and references therein). While these algorithms are fast and numerically stable, the running times typically scale as $O(nd^2)$ when the inputs have size $n \times d$ ($n \geq d$). With growing input sizes, these algorithms have become too slow to be practical. Often in applications, an approximate solution to these problems suffices. Hence, a major question is if we can trade off accuracy to obtain fast algorithms.

Frieze, Kannan and Vempala [FKV04] explored importance sampling based techniques to decrease the input sizes substantially so that the classical algorithms can then be run on the smaller inputs. Their work resulted in fast randomized approximation algorithms for the low rank approximation problem. Extending this work, [DMM06a, DMM06b, DRVW06, DV07] have studied applications of row sampling for matrix approximation.

In another line of work, Sarlós [Sar06] introduced the concept of structured oblivious subspace embeddings which can be used to decrease the input sizes so that the classical algorithms can now be run on the smaller inputs. For the least squares regression problem, Sarlós showed that one can obtain $1 + \varepsilon$ approximate solutions in faster than $O(nd^2)$ time that is required by the earlier algorithms to compute an exact solution. This led to numerous subspace embedding constructions such as [Tro11, CW17, NN13] leading to first input sparsity approximation algorithms for the low rank approximation and least squares regression problem.

These two lines of works have established sketching, in the form of sampling and randomized projection, as a key technique for obtaining fast approximate algorithms for classical linear algebra problems. In this thesis, we obtain new sketching algorithms that improve upon previous best algorithms for a number of problems in the classic setting.

The Streaming setting. In the classic setting described above, we assume that (i) the entire input is available to us and that (ii) the input fits in memory but that the classical algorithms are too slow when run directly run. In many settings, such as in internet traffic logs, financial transactions, the dataset is not given to us upfront but is defined by a series of updates to the underlying dataset. Representing the underlying dataset as an $n \times d$ matrix A, these updates typically take two forms: (i) **row arrival model**, in which the rows of the matrix A are revealed one after the other and (ii) **turnstile streaming model**, in which the individual entries (or entire rows) of the matrix A receive additive updates. When the parameters n and d are so large that the underlying matrix A cannot be maintained in memory as the updates arrive, we cannot use the algorithms from the classic setting to process such datasets.

The row-arrival setting is helpful to model time-series datasets wherein we receive new information at every time step and our underlying matrix is defined as the data that arrives in the next n time steps, where n is very large. It is also helpful to model the situations where the data is already available but does not fit in the memory. Suppose that there is a large dataset stored in the disk and the algorithms can only bring in rows of the dataset one after the other into the memory to operate on.

In such settings, we desire algorithms that use a small amount of memory and process the updates as they arrive to update their internal state and at the end of processing the stream, output an approximate solution to a problem on the underlying dataset. Such an algorithm, that uses much smaller than $n \cdot d$ space necessary to store the entire matrix A, can process long data streams defining a large underlying data matrix.

The first non-trivial streaming algorithm was proposed by Morris [Mor78] for (approximately) counting the number of items in a stream. In this problem, we see a stream of items and our objective is to simply count the number of items. We can trivially count the number of items using $\Theta(\log n)$ bits by simply incrementing a counter by 1, when the number of items in the stream is n. Perhaps surprisingly, Morris gave a randomized algorithm that uses only $\Theta(\log \log n)$ bits to approximate the number of items in the stream.

In 1996, Alon, Matias and Szegedy [AMS99] initiated the study of moment estimation in turnstile streams. In this problem, there is an underlying n dimensional vector x that receives updates of the form (i, Δ) in the stream. On receiving an update (i, Δ), the i-th coordinate x_i is updated as $x_i \leftarrow x_i + \Delta$. Our objective in this setting is to approximate $F_2(x) \coloneqq \sum_{i=1}^{n} |x_i|^2$. The so-called AMS sketch can approximate $F_2(x)$ up to a multiplicative $1 \pm \varepsilon$ factor using $O(\varepsilon^{-2})$ words of space. This work motivated a flurry of work [CCFC02, Cor05, IW05, Ind06, BJKS04, Li08, KNW10] studying algorithms and lower bounds for approximating $F_p(x) \coloneqq \sum_{i=1}^{n} |x_i|^p$ for $p \in [0, \infty)$. The frequency moment estimation problem has been a source of significant amount of techniques in the streaming and communication complexity literature.

In this thesis, we obtain improved streaming algorithms for problems such moment estimation, subspace approximation, top eigenvector approximation, etc., using the sketching techniques which convert the large underlying data matrix into a smaller form that can be efficiently updated on-the-fly as the updates to the underlying matrix arrive in the stream.

The Distributed setting. In many settings, the data we want to operate on is split across multiple servers. For example, some features of a user maybe stored on *Google Search* servers and the other set of features maybe stored on *YouTube* servers. Google may want to solve a regression problem or the low rank approximation problem on the matrix obtained by joining the data across multiple servers. Hence, a question is if such problems can be solved without communicating all the large matrices to a coordinator server. This is sometimes referred to as the **column partition** model.

In the **row partition** model, there are multiple servers each holding the data of a disjoint set of users. Again, one may want to solve problem such as linear regression or low rank approximation on the matrix obtained by combining the data of all the users. The question again is, if one can solve such problems without needing to send all the data to a coordinator servers.

The **arbitrary partition model** generalizes both of these models and allows for even more general data splits. In this model, there are s servers along with a coordinator and the j-th server holds an $n \times d$ matrix A_j and the coordinator wants to solve some problem on the matrix $A \coloneqq A_1 + \cdots + A_s$. In this model, [KVW14] studied communication efficient algorithms for the Principal Component Analysis (PCA) problem.

Apart from the **coordinator model** of distributed computation where the communication graph of the nodes is a *star*-graph, communication-efficient distributed algorithms have also been studied in more general graph topologies. One such model is the CONGEST model [Pel00] in which the data (such as a graph or rows of a matrix) is split across nodes and each node in the graph can send a limited amount of data to its neighbors in each round of communication. One then wants to solve a problem on the combined data using as few rounds of communication as possible.

In this thesis, we use sketching techniques to obtain communication efficient algorithms for solving the moment estimation problem, and more generally function sum estimation problem in the coordinator model, and we introduce the personalized CONGEST model which generalizes the CONGEST model and obtain communication efficient algorithms for regression and low rank approximation problems.

1.1 Measures of Efficiency

Each of the above settings presents different challenges with regard to the types of efficiency we care about. In this thesis, we consider five complexity measures with one or more of them being relevant to each of the above settings.

Time Complexity. In the classic setting, the time complexity simply refers to the total amount of time required by an algorithm to compute a solution to a problem on the given dataset.

In the streaming setting, we are concerned with the amount of time required by an algorithm to process an *update* to the underlying matrix. We call this the *update time* of a streaming algorithm. Having a small update time is crucial to be able to process a stream of updates at a high throughput. Another complexity measure related to time in the streaming setting is the amount of time required for a streaming algorithm to output a solution at the end of processing the stream.

Space Complexity. In the streaming setting, we assume that we do not have enough space to store the entire underlying matrix A and hence the algorithms operate in a memory-constrained setting. The space complexity of a streaming algorithm is the amount of space (in bits) it needs to process the stream of updates to the underlying matrix and output a solution at the end of the stream. We want algorithms that have as low a space complexity as possible.

Randomness Complexity. Many space and time efficient algorithms are often randomized and use a large number of random bits which raises the question how these bits are obtained. Thus it is important for an algorithm to use as few random bits as possible while meeting other efficiency measures.

More importantly though, in the streaming setting, an important question is how the large number of random bits needed by an algorithm are stored since the algorithms operate in a memory-constrained setting. The usual techniques here include replacing the use of full randomness with k-wise independent hash functions for some small value of k or use pseudorandom generators that fool small space algorithms, such as Nisan's PRG [Nis92].

Query Complexity. In many settings, specific ways of interacting with the underlying matrix can be much faster and simpler than materializing the entire matrix. Some ways of interacting with a matrix are (i) querying specific entries, or (ii) querying for the result of multiplying the matrix with a specific vector or more generally (iii) applying a function, chosen from a restricted class such as the set of all linear functions, to the matrix and obtaining the result.

In this setting, an algorithm needs to minimize the amount of queries it makes since it is directly related to the amount of time required to run such an algorithm. Thus given a class of queries that one is allowed, the query complexity of an algorithm is the number of such queries it performs to obtain a solution to the given problem. Again, we want algorithms that perform as few queries as possible.

Communication Complexity. In the distributed setting, the input to a problem can be arbitrarily partitioned among s servers. Assuming there is a coordinator that can communicate with all the servers, the communication complexity of a protocol is the number of bits exchanged between the coordinator and the s servers to solve the problem. We want protocols that use as few bits of communication as possible to solve a problem up to desired accuracy.

1.2 Classification of Techniques

In the randomized linear algebra literature, there have majorly been two different sketching techniques to obtain efficient algorithms: (i) linear sketching and (ii) coresets. We will briefly describe these two methods at a high level.

1.2.1 Linear Sketching

Linear sketching refers to the technique of applying an *oblivious* randomized linear transformation to decrease the dimensionality of the data or decrease the number of data points or both. At a high level, a sketching-based algorithm has the following structure: First the algorithm computes a sketch of the entire dataset using an appropriate transformation so that the sketch captures *enough* structure of the dataset sufficient to solve a problem and then using the sketch, the algorithm computes an (approximate) solution to the given problem on the original dataset.

To obtain fast sketch-based algorithms, it is necessary, as a first step that we can compute a sketch of the dataset quickly. After obtaining the sketch, we need an algorithm that runs quickly on the sketch to obtain a solution for the original problem. A proxy for how fast we can run an algorithm on the sketch is the so-called *size* of the sketch i.e., the output dimension of the randomized transformation. Thus, we need to carefully balance both the time-to-sketch and the size-of-the-sketch to obtain fast algorithms using this paradigm.

Sarlós [Sar06] observed that the Fast Johnson Lindenstrauss Transformation (FJLT) of Ailon and Chazelle [AC09] can be used to compute a sketch of an $n \times d$ matrix A in $O(nd \log n)$ time and showed that this sketch can be used to approximately solve problems such as the Frobenius norm Low Rank Approximation of A and the ℓ_2 linear regression problem with A as the coefficient matrix. Later, Clarkson and Woodruff [CW17] gave a construction of a sketch that can be computed in $\text{nnz}(A)$ time and obtained input-sparsity time algorithms for these problems. Thereafter, numerous works have obtained fast algorithms for a variety of problems using the sketch-and-solve paradigm.

Apart from solving problems in the classic setting, linear sketching is a major technique to obtain efficient algorithms in the streaming and distributed settings as well. We note that a linear sketch can be *efficiently*[1] updated when a coordinate of the underlying matrix A gets updated. This observation has lead to turnstile streaming algorithms for a variety of problems such as moment estimation [And17, KNPW11], heavy hitters [CCFC02], sampling coordinates from a variety of distributions [JST11, JW21] and many more.

In the arbitrary partition model of the distributed setting, linear sketches can be used to obtain communication efficient algorithms. Recall that each server in this setting holds a matrix $A(1), \ldots, A(s)$ respectively and if the server j sends a linear sketch of the matrix $A(j)$ to the coordinator, then the coordinator can obtain a sketch for the matrix $A = A(1) + \cdots + A(s)$ by simply *adding* up the sketches. When the *sketches* are small, this leads to communication-efficient algorithms in the coordinator model. This technique was used to solve the low rank approximation problem [KVW14].

1.2.2 Coresets

Coresets usually denote a weighted subset of the original data points such that solving an appropriately modified version of the original problem on this weighted subset leads to an approximate solution for the original problem. Again, to obtain fast algorithms using coresets, it is important that we are (i) able to compute the coreset quickly and (ii) the size of the coreset is small so that the problem can be solved quickly on the coreset.

In general, coresets are constructed using importance sampling of the data points where the

[1]In time independent of the size of the dataset.

importance of a point depends on the problem being solved and on how the data point interacts with rest of the points in the dataset. Sensitivity [FL11] is one way of assigning importance to each point in the dataset and sensitivities have been used to compute small coresets for a number of problems such as clustering [VX12], regression [DMM06b], etc.

We remark that while the sensitivities have been used to obtain coreset constructions for many problems, it is not always the case that they are the *best* measure of importance. For example, in the case of ℓ_p subspace embeddings, Cohen and Peng [CP15] show that importance sampling of rows using ℓ_p Lewis weights lead to coresets of smaller size as compared to coresets computed using the ℓ_p sensitivities.

Coresets have been used to obtain efficient algorithms for many problems in all the classic, streaming and distributed settings. In the classic setting, coresets are often used to reduce the size of the dataset to make searching for brute force solutions faster for problems such as clustering (see [CASS21] and references therein).

In the row-arrival model of streaming, coresets for many problems can be obtained using the "merge-and-reduce" framework [BS80]. Using this framework, any offline coreset construction algorithm can be converted into a streaming coreset construction in the row arrival model with only a space blow up of logarithmic factors in the length of the stream under certain conditions.

In the row-partition model of distributed computation, each server can independently compute a coreset for their data points and send it to the coordinator. In many constructions of coresets, simply the union of all the coresets is a coreset for the entire dataset.

1.3 Our Results in the Classic Setting

In this section, we will briefly describe the results obtained in this thesis for solving various problems in the classic setting.

1.3.1 Fast Oblivious Subspace Embeddings

Definition 1.3.1. A random matrix S with n columns is called an (α, δ) Oblivious Subspace Embedding (OSE) for d-dimensional subspaces of $\mathbb{R}^n$ if

$$\mathbf{Pr}_S[\text{for all } x \in W, \|x\|_2 \le \|Sx\|_2 \le \alpha\|x\|_2] \ge 1 - \delta$$

for all d-dimensional subspaces $W \subseteq \mathbb{R}^n$.

When δ is a small constant, we refer to S as an α-OSE for simplicity. We call α to be the *distortion* of the subspace embedding S. Instantiating W to be the column space of a given $n \times d$ matrix A, we obtain that with probability $\geq 1 - \delta$ over the random matrix $\boldsymbol{S}$, for all $x \in \mathbb{R}^d$,

$$\|Ax\|_2 \leq \|\boldsymbol{S}Ax\|_2 \leq \alpha\|Ax\|_2.$$

The concept of subspace embeddings was introduced by Sarlós [Sar06] who showed that it can be used to obtain fast algorithms for linear regression and low rank approximation, when $n \gg d$.

We can show, using a net argument, that if the entries of $\boldsymbol{S}$ are independent Gaussian random variables and if $\boldsymbol{S}$ has $m = O(d)$ rows, then $\boldsymbol{S}$ is an α-OSE for a constant α. But many applications of OSEs to linear algebra require that we are able to *quickly* compute the matrix $\boldsymbol{S} \cdot A$. If $\boldsymbol{S}$ is a dense Gaussian matrix, then unfortunately, it requires $\min(nd^{\omega-1}, \text{nnz}(A) \cdot d)$ time to compute the matrix $\boldsymbol{S} \cdot A$, where ω is the matrix multiplication constant, which is quite slow.

To remedy this problem, many structured OSEs [AC09, Tro11, CW17, NN13, Coh16] have been proposed in the literature that let us apply the subspace embedding $\boldsymbol{S}$ to a matrix A quickly. Using a combination of these structured OSEs, for a constant distortion α, and any $\gamma > 0$, we can obtain a random matrix $\boldsymbol{S}$ with $m = O(d \operatorname{polylog} d)$ rows and the matrix $\boldsymbol{S} \cdot A$ can be computed in time

$$O(\gamma^{-1}\,\text{nnz}(A) + d^{2+\gamma} \operatorname{polylog}(d)).$$

Before the work in this thesis, there was no construction of a subspace embedding that can be applied in $O(\gamma^{-1}\,\text{nnz}(A) + d^{2+\gamma} \operatorname{polylog}(d))$ time and with $m = o(d \log d)$ rows. In this thesis, we obtain the *first* construction of an OSE with $m = O(d \cdot \operatorname{poly}(\log\log d))$ rows and a distortion $\alpha = \exp(\operatorname{poly}(\log\log d))$ such that the matrix $\boldsymbol{S} \cdot A$ can be computed in

$$O(\gamma^{-1}\,\text{nnz}(A) + d^{2+\gamma} \operatorname{polylog}(d))$$

time. This leads to the first algorithms for many problems, such as basis finding, that run in time $O(\text{nnz}(A) + d^{\omega}(\operatorname{poly}(\log\log d)))$ making steps towards completely removing the $\operatorname{polylog}(d)$ multiplicative factor to the d^{ω} term plaguing many results before our work. We will briefly summarize our main result in the following theorem.

Theorem 1.3.2. *Given a parameter $\gamma > 0$, there is an $\exp(\operatorname{poly}(\log\log d))$-OSE $\boldsymbol{S}$ with $m = O(d \cdot \operatorname{poly}(\log\log d))$ rows such that given an $n \times d$ matrix A, we can compute the matrix $\boldsymbol{S} \cdot A$ in time*

$$O(\gamma^{-1}\,\text{nnz}(A) + d^{2+\gamma} \operatorname{polylog} d).$$

Our crucial observation which led to this result is that when all the *unit* vectors in a subspace have large ℓ_1 norms, then a sparse sign matrix satisfies the OSE property. We use a deterministic embedding construction of Indyk [Ind07] and show that any d-dimensional subspace of $\mathbb{R}^{O(d \operatorname{polylog}(d))}$ can be linearly mapped to such a flat subspace without blowing up the dimension by a lot.

We note that [CDDR23] have recently proposed a new sparse OSE construction which improves upon the results in this thesis.

1.3.2 Approximating Sum-of-Distances

Consider the following simple problem: given n points $x_1, \ldots, x_n \in \mathbb{R}^d$ and a shape $S \subseteq \mathbb{R}^d$, we want to approximate

$$\sum_{i\in[n]} d(x_i, S)$$

where $d(x, S)$ is defined as $\inf_{y\in S} \|x - y\|_2$. When S is a set of k points, the above quantity is just the k-median cost with S as the centers and when S is a k-dimensional subspace, then the quantity is the ℓ_1 subspace approximation cost. For this problem, Sohler and Woodruff [SW18] gave a dimensionality reduction by producing a subspace P of dimension $\text{poly}(k/\varepsilon)$ such that for any subset $S \subseteq \mathbb{R}^d$ with $\dim(\text{span}(S)) \le k$,

$$\sum_{i\in[n]} \sqrt{d(x_i, P)^2 + d(\mathbb{P}_P x_i, S)^2} = (1 \pm \varepsilon) \sum_{i\in[n]} d(x_i, S),$$

where $\mathbb{P}_P$ denotes the orthogonal projection matrix on to subspace P. The above relation shows that one only needs the projections of x_is onto this special subspace P and the distance from x_i to P to be able to approximate the sum-of-distances to an arbitrary k-dimensional shape. Thus, we only need to store $n \cdot \text{poly}(k/\varepsilon)$ parameters instead of the $n \cdot d$ parameters required to store the exact values of x_is, to be able to approximate the sum-of-distances. Unfortunately, the construction in [SW18] takes time exponential in k/ε. In this thesis, we give an improved algorithm that computes such a subspace P in time $\widetilde{O}(\mathsf{nnz}(A)/\varepsilon^2 + (n + d)\,\text{poly}(k/\varepsilon))$ and the reduced dimension points can be used to approximate the sum of distances of the original n points to any k-dimensional shape. Our algorithm is iterative and in each iteration solves a subspace approximation problem using linear sketching techniques to reduce the *size* of the problem.

The dimensionality reduction procedure also lets us compute small coresets for many *sum-of-distances* problems. We summarize our main result in the following theorem.

Theorem 1.3.3. *Given an $n \times d$ matrix A, a rank parameter k and an accuracy parameter ε, there is an iterative algorithm that runs in time $O(\text{nnz}(A)/\varepsilon^2 + (n + d) \cdot \text{poly}(k/\varepsilon))$ and outputs a subspace P of dimension at most $\text{poly}(k/\varepsilon)$ such that with probability $\ge 9/10$, for any shape S with $\dim(\text{span}(S)) \le k$,*

$$\sum_{i\in[n]} \sqrt{d(A_{i,*}, P)^2 + d(\mathbb{P}_P \cdot A_{i,*}, S)^2} = (1 \pm \varepsilon) \sum_{i\in[n]} d(A_{i,*}, S),$$

where $A_{i,*} \in \mathbb{R}^d$ denotes the i-th row of the matrix A and $\mathbb{P}_P$ is the orthogonal projection matrix onto the subspace P.

1.3.3 Query Complexity of Low Rank Approximation

Very often it is much more efficient to interact with the underlying matrix in specific ways than it is to assume arbitrary access to the entries of the matrix. For example, suppose we are given access to an $n \times n$ matrix A, when the actual matrix we want to operate on is A^q where q is an integer ≥ 1. Given a vector x, we can compute $A^q x$ using q adaptive matrix-vector products whereas computing the entire matrix A^q is inefficient. Thus given the matrix A, we have *efficient* matrix-vector product query access to A^q. Matrix-vector products model a specific set of linear functions on matrices. We can indeed allow for more general linear measurements of matrices and study the query complexity in those models.

Candés and Plan [CP11] study the number of *noisy* linear measurements required to extract an underlying $n \times n$ low rank matrix A. Here a linear measurement is defined as a linear transformation mapping $n \times n$ matrices to real numbers. Note that an $n \times n$ rank-k matrix A can be described using only $O(nk)$ parameters. They give algorithms with query complexity that increases with the noise level in the measurements. At a certain large enough noise level, they show that their algorithm requires $\Omega(n^2)$ linear measurements which amounts to essentially reading all the entries of the matrix A and hence could be inefficient to implement. The algorithms they study are non-adaptive, as in, they specify all the linear measurements up front and reconstruct the matrix using the received responses. Hence, a natural question is if adaptivity helps us reconstruct the matrix with fewer number of linear measurements.

Indeed, we can show that using the power method, for the noise level for which the algorithm of [CP11] requires $\Omega(n^2)$ linear measurements, we can reconstruct the matrix using a total of $\widetilde{O}(n)$ linear measurements and $O(\log n)$ rounds of adaptivity. But are $\Omega(\log n)$ rounds of measurements required to avoid essentially reading the whole matrix? Unfortunately, this turns out to be (almost) true. In this thesis, we show that any algorithm that runs for fewer than $o(\log n/\log\log n)$ adaptive rounds, must use a total of $\Omega(n^2)$ linear measurements and therefore essentially has to read all the entries of the matrix [KW23].

Our lower bound is proved using Bayes risk framework [CGZ16]. We define a distribution over $n \times n$ matrices such that any *deterministic* algorithm that uses $n^{2-\beta}$ linear measurements in each round does not learn enough information in $o(\log n/\log\log n)$ rounds to output a good rank-k approximation. By Yao's lemma, it then follows that there is no randomized algorithm that outputs a low rank approximation for every input. Our input distribution is simply $\boldsymbol{G} + (\alpha/\sqrt{n})\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ where $\boldsymbol{G}$ is an $n \times n$ matrix with independent Gaussian entries, $\boldsymbol{u}_i, \boldsymbol{v}_i$ for $i = 1, \ldots, k$ are n-dimensional vectors with independent Gaussian entries as well, and α is a large enough constant. We summarize the main result in the following theorem:

Theorem 1.3.4 (Informal)**.** *There exists a constant c such that any randomized algorithm which makes $m \geq nk$ noisy linear measurements of an arbitrary rank-r matrix A with $\|A\|_{\mathrm{F}} = \Theta(nk)$ in each of t rounds, and outputs an estimate $\hat{A}$ satisfying $\|A - \hat{A}\|_{\mathrm{F}}^2 \leq c\|A\|_{\mathrm{F}}^2$ with probability $\geq 9/10$ over the randomness of* *the algorithm and the Gaussian noise, requires $t = \Omega(\log(n^2/m)/(\log\log n))$.*

Using this result we also show that general linear measurements are not more powerful than matrix-vector products for problems such as low rank approximation, eigenvalue approximation etc.

1.3.4 Reduced-Rank Regression with Operator Norm Error

In the same vein as above sketch-based algorithms, we obtain a fast algorithm for approximately solving

$$\min_{X:\operatorname{rank}(X)\le k} \|AX - B\|_2.$$

Note that the usual multi-response regression problem measures the error in terms of the Frobenius norm and has no restriction on the rank of the coefficient matrix. In the version of the problem we study, we put a rank restriction to increase interpretability of the coefficient matrix and to decrease the number of free parameters, and also use operator norm of the residual matrix to measure the error which prefers solutions X for which the residual error is more spread out compared to the Frobenius norm solution.

The rank restriction combined with a lack of *Pythagorean Theorem* equivalent for operator norm makes the problem harder to solve. We do not know of a closed form solution to this problem. We use a criterion of Sou and Rantzer [SR12] for there to exist a solution to the problem with $\|AX - B\|_2 \le \beta$ for a given β. We then show that a rank-k approximation for a specific matrix in operator norm gives a solution to the reduced-rank regression problem.

To obtain a fast algorithm, we also show that the Block Krylov iteration algorithm of [MM15] works even when the matrix-vector products have a certain amount of error. We summarize our result in the following theorem:

Theorem 1.3.5. *Given an $n \times d$ matrix A, an $n \times d'$ matrix B, an accuracy parameter ε and a rank parameter k, there is an algorithm that runs in time*

$$O\left(\left(\frac{\operatorname{nnz}(A)\cdot k}{\varepsilon^{3/2}} + \frac{\operatorname{nnz}(B)\cdot k}{\varepsilon} + \frac{d^2 k}{\varepsilon^{3/2}}\right)\cdot \operatorname{polylog}(\kappa(B), n, d, 1/\varepsilon) + d^{\omega}\right)$$

where $\kappa(B) = \sigma_1(B)/\sigma_{k+1}(B)$ is the rank-k condition number of the matrix B.

1.3.5 Ridge Regression

Given a matrix $A \in \mathbb{R}^{n\times d}$ and a vector $b \in \mathbb{R}^d$, the ridge regression problem is defined as:

$$\min_x \|Ax - b\|_2^2 + \lambda\|x\|_2^2$$

where $\lambda > 0$ is a *regularization* parameter. When $n \geq d$, then we say we are in the *over-determined* case and when $n < d$, we say we are in the *under-determined* case. We study fast algorithms for computing a vector $\widetilde{x} \in \mathbb{R}^d$ that satisfies

$$\|A\widetilde{x} - b\|_2^2 + \lambda\|\widetilde{x}\|_2^2 \leq (1+\varepsilon) \min_x (\|Ax - b\|_2^2 + \lambda\|x\|_2^2). \tag{1.1}$$

In the ridge regression objective above, as we increase the value of λ, the *importance* of the so-called design matrix A decreases, as in, when λ is large enough, setting $x = 0$ without considering A and b would be a good solution. Thus, the *effective dimension* of the problem decreases with increasing values of λ. To capture this phenomenon, we define statistical dimension sd_λ as

$$\mathsf{sd}_\lambda \coloneqq \sum_{i=1}^{\min(n,d)} \frac{\sigma_i^2}{\sigma_i^2 + \lambda},$$

where σ_i is the i-th singular value of the matrix A. In general, we would like algorithms for solving ridge regression to have small running times when sd_λ is small since we already know that $x = 0$ is a *good* solution.

Fast Algorithms in the Under-determined Case

In this thesis, we study algorithms for Ridge Regression in the underdetermined case [KW22]. Earlier algorithms [CYD18] used Oblivious Subspace Embeddings with distortion $1+\varepsilon$ to obtain a solution with the guarantee in (1.1). We show that a sketch with only the weaker ε-Approximate Matrix Multiplication (AMM) guarantee and an Oblivious Subspace Embedding guarantee with a constant distortion is sufficient to solve the ridge regression problem to a $1+\varepsilon$ factor.

Definition 1.3.6 (AMM)**.** A sketch $\boldsymbol{S}$ with n columns has the (ε, δ)-AMM property if given any two matrices A and B each with n rows,

$$\mathbf{Pr}_{\boldsymbol{S}}[\|(\boldsymbol{S}A)^{\mathrm{T}}(\boldsymbol{S}B) - A^{\mathrm{T}}B\|_{\mathrm{F}} \leq \varepsilon\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}] \geq 1 - \delta.$$

Our observation that only constant distortion OSEs which additionally satisfy AMM guarantees is sufficient to solve ridge regression leads to faster algorithms in some parameter regimes compared to [CYD18]. We also give a tight lower bound on sizes of the sketches that have AMM guarantee.

Theorem 1.3.7 (AMM lowerbound)**.** *Given an accuracy parameter ε and an integer n, any randomized sketching matrix $\boldsymbol{S}$ with n columns that satisfies*

$$\mathbf{Pr}_{\boldsymbol{S}}[\|(\boldsymbol{S}A)^{\mathrm{T}}\boldsymbol{S}B - A^{\mathrm{T}}B\|_{\mathrm{F}} \leq \varepsilon\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}] \geq 9/10,$$

for all matrices A, B with n rows, must have $m = \Omega(\min(n, 1/\varepsilon^2))$ rows.

We note that the above lower bound is tight up to constant factors since the CountSketch matrix with $m = O(1/\varepsilon^2)$ rows satisfies the ε-AMM guarantee.

Optimal Deterministic Coresets

We study the problem of constructing coresets for the ridge regression problem [KW20]. Specifically, given an instance of the ridge regression problem (A, b, λ), we want to compute a set $S \subseteq [n]$ and weights w_i for $i \in S$ such that the solution to the problem

$$\min_x \sum_{i \in S} w_i(\langle A_{i,*}, x\rangle - b)^2 + \lambda\|x\|_2^2 \tag{1.2}$$

is a $1 + \varepsilon$ approximation to the ridge regression problem. We prove the following theorem:

Theorem 1.3.8 (Informal)**.** *Given any ridge regression instance (A, b, λ) and an accuracy parameter ε, there is a deterministic algorithm to find a subset $S \subseteq [n]$, $|S| = O(\mathsf{sd}_\lambda/\varepsilon)$ with corresponding weights $(w_i)_{i \in S}$ such that the optimal solution for (1.2) is a $1 + \varepsilon$ approximation for the ridge regression problem (A, b, λ).*

We additionally show that this bound cannot be improved in terms of statistical dimension by exhibiting an instance for every ε and $\lambda \le O(1/\varepsilon)$ which requires that any such coreset have $\Omega(\mathsf{sd}_\lambda/\varepsilon)$ rows.

Using our deterministic coreset, we also give communication efficient algorithms for ridge regression in the row-partition model of distributed computation.

1.3.6 Linear-time Attention Mechanism via Sketching Polynomial Kernels

The softmax attention mechanism with causal masking [VSP+17] is the key ingredient behind the success of large language models. Let Q, K and V be arbitrary $n \times h$ matrices. Let $q_1, \ldots, q_n \in \mathbb{R}^h$, $k_1, \ldots, k_n \in \mathbb{R}^h$ and $v_1, \ldots, v_n \in \mathbb{R}^h$ be the rows of the matrices Q, K and V respectively. The output of the softmax attention mechanism with causal masking is defined to be the $n \times h$ matrix O with the i-th row o_i defined as

$$o_i \coloneqq \sum_{j \in [i]} \frac{\exp(\langle q_i, k_j\rangle/\sqrt{h})}{\sum_{j' \in [i]} \exp(\langle q_i, k_{j'}\rangle/\sqrt{h})} \cdot v_j. \tag{1.3}$$

A naive implementation of the attention mechanism involves computing the $n \times n$ matrix $\exp(Q \cdot K^{\mathrm{T}}/\sqrt{h})$, where $\exp(\cdot)$ denotes entrywise exponentiation. Such an implementation takes $O(n^2 h)$ time limiting the value of n that can be used. Thus, it is very important to decrease the time complexity of the attention mechanism. Unfortunately, Alman and Song [AS23a] show that obtaining accurate entrywise approximation to the matrix O requires $\Omega(n^2)$ time, assuming the Strong Exponential Time Hypothesis (SETH). Thus, reasonable approximations to the matrix O in $o(n^2)$ time seem hard to obtain.

In this thesis, we explore the use of even degree polynomials instead of $\exp(\cdot)$ in the attention mechanism. Concretely, we propose polynomial attention mechanism, where the output is the $n \times h$ matrix O', with i-th row o'_i defined as

$$o'_i \coloneqq \sum_{j \in [i]} \frac{\langle q_i, k_j \rangle^p}{\sum_{j' \in [i]} \langle q_i, k_{j'} \rangle^p} \cdot v_j.$$

Via extensive experiments, we argue that polynomial attention with even degree $p \geq 4$ is an effective replacement for the softmax attention (1.3). The advantage of replacing exponentials with a degree p polynomial is that the matrix O' can now be computed exactly in time $O(n \cdot h^{p+1})$ using the fact that for any vectors q and k, we have $\langle q^{\otimes p}, k^{\otimes p} \rangle = \langle q, k \rangle^p$, where $q^{\otimes p}$ denotes the h^p dimensional vector obtained by tensoring q with itself p times. To implement the mechanism in $O(nh^{p+1})$ time, we first compute the $n \times h^p$ matrices $Q^{\otimes p}$ and $K^{\otimes p}$ obtained by tensoring each of the rows of Q and K for p times, and we then use a simple prefix sum based algorithm to compute the matrix O'.

While the time complexity of polynomial attention is linear in n, for typical values of h such as 64, 128 and 256, even for $p = 4$, the h^{p+1} factor is quite large and hence the number of columns in the matrices $Q^{\otimes p}$ and $K^{\otimes p}$ are quite large. To mitigate this issue, we use sketches for the polynomial kernel from [AKK⁺20] to compute $n \times h'$ matrices $Q' = Q^{\otimes p} \cdot \boldsymbol{S}$ and $K' = K^{\otimes p} \cdot \boldsymbol{S}$ for $h' \ll h^p$ such that

$$Q' \cdot (K')^{\mathrm{T}} \approx Q^{\otimes p} \cdot (K^{\otimes p})^{\mathrm{T}}.$$

We can then use the matrices Q' and K' instead of $Q^{\otimes p}$ and $K^{\otimes p}$ and compute an approximation for the polynomial attention output O'. A key advantage of the sketch $\boldsymbol{S}$ of [AKK⁺20] is that the matrices $Q^{\otimes p} \cdot \boldsymbol{S}$ and $K^{\otimes p} \cdot \boldsymbol{S}$ can be computed without computing the matrices $Q^{\otimes p}$ and $K^{\otimes p}$. Therefore, the sketches Q' and K' can be computed much faster than $O(n \cdot h^p \cdot h')$ time. Using additional techniques such as self-tensoring to ensure non-negativity of dot products, and a new block-based algorithm for causal masking, we show that the approximate polynomial attention mechanism is much faster than the vanilla softmax attention mechanism on ML accelerators such as GPUs and TPUs, and that the approximate polynomial attention fares well against softmax attention.

1.4 The Streaming Setting

1.4.1 Turnstile Streaming

As we mentioned before, in the turnstile streaming model, the entries of the underlying matrix receive additive updates in the stream. Concretely, the underlying matrix A receives updates of the form $((i, j), \Delta)$ and on receiving such an update, we update the (i, j)-th entry as $A_{i,j} \leftarrow A_{i,j} + \Delta$.

Here Δ is allowed to take on negative values as well. The goal of a streaming algorithm is to process the stream of updates in as small space as possible and at the end of the stream output a solution to a problem on the underlying matrix.

There has been a lot of work studying optimal space bounds for various problems when the underlying object is an n-dimensional vector x receiving updates of the form (i, Δ). For the problem of approximating $F_k(x) = \sum_{i=1}^{n} |x_i|^k$, up to constant factors, an $\Omega(n^{1-2/k})$ bits lower bound was shown by [BYJKS04] for $k > 2$ and a matching upper bound (up to polylogarithmic factors) was given by [IW05]. Much later, Andoni [And17] also gave a simple linear sketch based algorithm to approximate $F_k(x)$ using $\widetilde{O}(n^{1-2/k})$ bits of space.

When $k \le 2$, a series of works ending with [KNW10] give a turnstile streaming algorithm that uses $O(\varepsilon^{-2} \log n)$ bits to approximate $F_k(x)$ up to $1 \pm \varepsilon$ factors. They also show that $\Omega(\varepsilon^{-2} \log n)$ bits are necessary thus completely resolving the space complexity of estimating $F_k(x)$ for $k \le 2$.

While the space complexity of estimating $F_k(x)$ in a turnstile stream has been resolved, unfortunately the *update time* of these space efficient algorithms is quite large. Andoni's algorithm for estimating $F_k(x)$ for $k > 2$ uses a pseudorandom generator of Nisan and Zuckerman [NZ96] to achieve a space complexity of $O(n^{1-2/k} \operatorname{poly}(\log n))$ bits and therefore has an update time of $\operatorname{poly}(n)$, which makes the algorithm impractical. For $k \le 2$, the algorithm of [KNW10] has an update time of $\operatorname{poly}(1/\varepsilon)$ in the WordRAM model. A later work [KNPW11] gave a new algorithm that still uses the optimal $O(\varepsilon^{-2} \log n)$ bits but has an improved update time of $O(\log^2 1/\varepsilon \log\log 1/\varepsilon)$ in the WordRAM model with a word size $\Omega(\log n)$.

As we noted before, algorithms that have a small update time are necessary to be able to handle high-throughput streams. Thus, the main question is to obtain space-optimal algorithms that also have a small update time.

To explain why the update time of these algorithms is large, we will first give the simple recipe many streaming algorithms follow. First the streaming algorithms define a randomized sketching matrix $\boldsymbol{S}$ with d columns. Each of the columns of $\boldsymbol{S}$ is sampled *independently* from an appropriate distribution. The streaming algorithm essentially maintains the value of $y = \boldsymbol{S}x$ when the vector x is being updated in the stream as follows: when the vector x gets an update (i, Δ), the streaming algorithm retrieves $\boldsymbol{S}_{*,i}$, the i-th column of $\boldsymbol{S}$ and updates $y \leftarrow y + \boldsymbol{S}_{*,i} \cdot \Delta$. Note that this will ensure that at the end of processing the stream $y = \boldsymbol{S}x$ where x is the final value of the underlying vector. Then the algorithm uses the vector y to output a solution to the problem it is trying to solve.

In the above recipe, the streaming algorithm needs to be able to retrieve any column of the matrix $\boldsymbol{S}$. As the columns are all independently sampled, the algorithms thus require $\Omega(n)$ bits of space to store the randomness used to sample the columns of $\boldsymbol{S}$. There have majorly been two ways to address this problem: (i) Instead of sampling the columns of $\boldsymbol{S}$ independently, use a t-wise independent hash function, for an appropriate value of t, to define the matrix $\boldsymbol{S}$ or (ii) Use a pseudorandom generator (PRG) that fools small-space algorithms to define the matrix $\boldsymbol{S}$ and only store the *seed* for the pseudorandom generator so that the any column of $\boldsymbol{S}$ can be generated on demand.

[KNW10, KNPW11] use the first route whereas [And17] uses the second route, which was first suggested by Indyk [Ind06]. When the amount of independence, t, required is large, the hash functions drawn from t-wise independent hash families are slow to evaluate. Similarly, when we are trying to fool algorithms using pseudorandom generators, either the hash functions have to be evaluated on large inputs or many hash functions have to be evaluated sequentially.

In this thesis [KPTW23], we generalize the construction of Nisan's PRG [Nis92] and give a new pseudorandom generator, which we call HashPRG, that has a space-vs-time trade-off, using which we can obtain fast update times by using more space. Our construction additionally has a symmetry property that lets us derandomize the guarantees Minton and Price [MP14] give for a fully random CountSketch data structure. Using HashPRG and opening up the analysis of the algorithms of Andoni [And17], we obtain a constant factor F_k approximation algorithm for $k > 2$ that uses $\widetilde{O}(n^{1-2/k})$ bits of space and has an $O(1)$ update time in the Word RAM model.

Similarly, for $k < 2$ and $\varepsilon < 1/n^c$, we obtain an algorithm based on [KNPW11] to approximate $F_k(x)$ up to a $1 \pm \varepsilon$ factor using the optimal $O(\varepsilon^{-2} \log n)$ bits of space and an update time of $O(\log n)$ in the Word RAM model. We also obtain algorithms with fast update times for other problems such as for estimating $\|x\|_\infty$. We summarize some of our results in the following theorem:

Theorem 1.4.1. *Suppose that a vector $x \in \mathbb{R}^n$ initialized to 0 receives the updates $(i_1, \Delta_1), \ldots, (i_t, \Delta_t)$ and suppose that the number of updates $t \le \text{poly}(n)$ and $|\Delta_j| \le \text{poly}(n)$ for all j. On a WordRAM machine with a word size of $\Omega(\log n)$, the following results hold:*

1. *For $k > 2$, there is a turnstile streaming algorithm to approximate $F_k(x)$ up to a factor dependent on k, using $\widetilde{O}(n^{1-2/k})$ bits of space. The algorithm has an update time of $O(1)$.*
2. *For $k < 2$ and $\varepsilon < 1/n^c$ for a small enough constant c, there is a turnstile streaming algorithm that uses the optimal $O(\varepsilon^{-2} \log n)$ bits of space and approximates $F_k(x)$ up to a multiplicative $1 \pm \varepsilon$ factor. The algorithm has an update time of $O(\log n)$.*
3. *There is a turnstile streaming algorithm that uses the optimal $O(\varepsilon^{-2} \log 1/\varepsilon \log n)$ bits of space and outputs an additive $\varepsilon\|x\|_2$ approximation to $\|x\|_\infty = \max_i |x|_i$. The algorithm has an $O(\log 1/\varepsilon)$ update time.*
4. *There is a turnstile data structure that uses $O(tr \log n + \log^2 n)$ bits of space and a deterministic algorithm that given any $i \in [n]$ at the end of the stream, uses the state of the data structure, to obtain an estimate $\hat{x}_i$ such that for all i,*

$$\mathbf{Pr}[|x_i - \hat{x}_i| > \alpha\|x\|_2/\sqrt{t}] \le 2\exp(-\alpha^2 r) + 1/\text{poly}(n).$$

The algorithm has an update time of $O(r \log n)$.

1.4.2 Row Arrival Model

Recall that in the row-arrival model of streaming, we obtain rows of the matrix A one after the other. Using a small amount of space, we want to solve some problem on the underlying matrix A. In [EKM$^+$24b], we give fast streaming algorithms to compute coresets for the ℓ_∞ subspace approximation problem defined as

$$\min_{\text{dim-}k \text{ subspaces } V} \max_i d(A_{i,*}, V),$$

which essentially asks us to find the k-dimensional subspace V that minimizes the maximum distance from the points in the matrix A. Our key coreset construction result is summarized below:

Theorem 1.4.2 (Informal). *Given rows of an $n \times d$ matrix A in a stream, there is a deterministic coreset construction algorithm that uses at most the space required to store $O(k \cdot \log^2(n\kappa))$ rows and computes a subset of rows $S \subseteq [n]$ such that for any k-dimensional subspace V,*

$$\frac{\max_{i \in [n]} d(A_{i,*}, V)}{\sqrt{k} \log(n\kappa)} \le \max_{i \in S} d(A_{i,*}, V) \le \max_{i \in [n]} d(A_{i,*}, V).$$

In the above theorem, κ is a suitably defined *online rank-k condition number*. Given that the entries of the matrix A are integers bounded in absolute value by $\mathrm{poly}(n)$, we can show that $\kappa \le n^{O(k)}$. The above theorem then implies that an approximate solution to the ℓ_∞ subspace approximation problem on the rows $(A_{i,*})_{i \in S}$ is also an approximate solution to the ℓ_∞ subspace approximation problem on the entire matrix A.

We also provide an almost matching lower bound instance showing that it is impossible to obtain a coreset with distortion $\le \sqrt{k/\log n}$ using $o(n)$ bits of space.

We show applications of our coreset construction for many problems such as ℓ_p subspace approximation, width estimation, volume estimation, etc. Our work is the first to construct such coresets for general values of k, whereas the previous works studied streaming algorithms only for the specific values such as $k = 0$ and $k = 1$ [AS15].

1.4.3 Approximating the Top Singular Vector

In the row-arrival model, we study the problem of approximating the top singular vector of the matrix A. Suppose that we receive the rows $a_1, \ldots, a_n \in \mathbb{R}^d$ one after the other. To compute the top right singular vector of A, we can simply maintain the $d \times d$ matrix $\sum_i a_i a_i^{\mathrm{T}}$ in the stream using $\widetilde{O}(d^2)$ bits of space and then the top eigenvector of the PSD matrix $\sum_i a_i a_i^{\mathrm{T}}$ is then the top right singular vector of the matrix A.

Thus, the main question is if we can approximate the top eigenvector using $o(d^2)$ bits of space. In particular, what can we do using $\widetilde{O}(d)$ bits of space i.e., the space enough to store only $\widetilde{O}(1)$ rows of the matrix A. Price [Pri23] showed that when the spectral gap $\sigma_1(A)/\sigma_2(A) \le O(1)$, then any streaming algorithm must use $\widetilde{\Omega}(d^2)$ bits of space to approximate the top singular vector whereas when $\sigma_1(A)/\sigma_2(A) \ge C\sqrt{\log n \log d}$, then there is a streaming algorithm that uses $\widetilde{O}(d)$ bits of space. The gap requirement for Price's algorithm to work is quite large but an advantage of Price's algorithm is that it works for arbitrary order streams. We relax the arbitrary order assumption and assume that the rows of the matrix A are revealed to us in a uniformly random order and show that we can obtain an approximation to the top singular vector using $\widetilde{O}(d)$ bits of space when the gap $\sigma_1(A)/\sigma_2(A) > C$ if the stream does not have any *heavy* rows.

We also obtain an algorithm that uses $O(t \cdot d \cdot \mathrm{polylog}(d))$ bits of space when t is the number of *heavy* rows defined as the rows with a euclidean norm larger than $\|A\|_{\mathrm{F}}/(\sqrt{d} \cdot \mathrm{polylog}(d))$. In the vein of Price's analysis, we show that using $\Omega(td)$ bits of space is necessary to obtain an accurate top singular vector approximation when the gap is a constant, even in random order streams, thus showing parameterizing in terms of the heavy rows is necessary to obtain accurate approximations of the top singular vector. We summarize our main result in the following theorem:

Theorem 1.4.3. *Let A be an arbitrary $n \times d$ matrix with a gap $R = \sigma_1(A)^2/\sigma_2(A)^2$ and let t be the number of heavy rows in A i.e., the rows with a Euclidean norm larger than $\|A\|_{\mathrm{F}}/(\sqrt{d} \cdot \mathrm{polylog}(d))$. There is a randomized streaming algorithm such that when the rows of matrix A are given to the algorithm in a uniform random order, the algorithm uses $O(\max(1, t) \cdot d \cdot \mathrm{polylog}(n))$ bits of space and outputs a unit vector $\hat{v}$ that satisfies*

$$\langle \hat{v}, v_1 \rangle^2 \ge 1 - \frac{1}{C\sqrt{R}}$$

with probability $\ge 9/10$. Here $v_1 \in \mathbb{R}^d$ is the top right singular vector of the matrix A.

1.5 The Distributed Setting

1.5.1 Function Sum Approximation

Consider the arbitrary partition model. In this setting, there are s servers all connected to a coordinator via bidirectional communication links. The j-th server holds a matrix $A(j) \in \mathbb{R}^{n \times d}$ and the coordinator wants to solve a problem on the matrix $A = A(1) + \cdots + A(s)$. In each round of communication, each of the servers can send a message to the coordinator and based on all the messages received the coordinator can send a possibly distinct message to each of the servers. A protocol can use multiple such rounds of communication to solve a problem. The amount of communication of a protocol is the total number of bits of communication sent/received by the coordinator.

In this thesis [EKM+24a], we study the problem of moment estimation and more generally arbitrary function sum approximation problem. In this problem, the j-th server holds a non-negative vector $x(j) \in \mathbb{R}^n$ and the coordinator wants to approximate $\sum_i f(x_i)$, where x_i is the i-th coordinate of the vector $x = x(1) + \cdots + x(s)$ and f is an arbitrary nonnegative function. If $f(y) = y^k$, then the coordinator simply wants to approximate the F_k moment[2] of vector x. We give a two round protocol for this problem when the function f is super-additive and satisfies a few other properties. To capture the communication complexity of the problem, we define a parameter $c_f[s]$ which is the smallest value that satisfies

$$f(y_1 + \cdots + y_s) \le \frac{c_f[s]}{s}(\sqrt{f(y_1)} + \cdots + \sqrt{f(y_s)})^2 \text{ for all } y_1, \ldots, y_s \ge 0.$$

We define a certain "approximate invertibility" property and our protocol approximates $\sum_i f(x_i)$ that satisfies the approximate invertibility property. At a high level, approximate invertibility requires that given a multiplicative approximation to x, we can obtain a multiplicative approximation to $f(x)$ and vice-versa. Our result is summarized in the following theorem.

Theorem 1.5.1 (Informal). *Let f be an arbitrary super-additive nonnegative function that satisfies approximate invertibility. Let each of the s servers hold a nonnegative vector $x(1), \ldots, x(s)$ respectively. In the coordinator model, there is a two round protocol that uses $O(c_f[s] \cdot \mathrm{polylog}(n)/\varepsilon^2)$ bits of total communication and approximates $\sum_i f(x_i)$ up to $(1 \pm \varepsilon)$ factor with probability $\ge 9/10$, where $x = x(1) + \cdots + x(s)$.*

Specifically, for estimating $\sum_i x_i^k$ for $k > 2$, we can show that $c_f[s] = s^{k-1}$ and therefore our algorithm uses a total of $O(s^{k-1} \mathrm{polylog}(n)/\varepsilon^2)$ bits of communication. This matches the $\Omega(s^{k-1}/\varepsilon^2)$ lower bound of [WZ12]. We additionally show a lower bound of $\Omega(s^{k-1}/\varepsilon^k)$ bits on the amount of communication that any one-round protocol must use therefore showing that our protocol achieves the best communication bounds using minimum possible rounds.

For general functions f satisfying certain properties, we also show an $\Omega(c_f[s]/\varepsilon^2)$ lower bound on the amount of communication thus showing that our protocol achieves near-optimal communication bounds.

1.5.2 Personalized CONGEST model

As discussed earlier, in the coordinator model, a server can only talk with the coordinator and hence the communication graph is a *star* graph. We define the personalized CONGEST model which supports arbitrary graph topologies. Let the topology of the servers be defined by a $G = (V, E)$, where V is the set of servers and E denotes the set of pairs of servers that can communicate with each other. Let Δ denote a distance parameter. Suppose that each server v holds a matrix A_v, define a matrix $A^{(\Delta,v)}$ formed by the union of the rows of all the matrices present at servers at a distance at most Δ in the graph G. Each server now simultaneously wants to solve a regression problem or low rank approximation problem on the matrix $A^{(\Delta,v)}$.

We obtain communication efficient algorithms for solving these problems using only Δ rounds of communication. Our result is summarized in the following theorem:

[2]Recall $F_k(x) \coloneqq \sum_i |x_i|^k$.

Theorem 1.5.2 (Informal)**.** *Let $G = (V, E)$ be the graph with servers as nodes and communication links as edges. Let Δ be a distance parameter. Let $A_v \in \mathbb{R}^{n_v \times d}$ be a matrix held by server v and $A^{(\Delta,v)}$ be the matrix formed by the union of rows in a distance Δ neighborhood of v in the graph G. There is a Δ round protocol, wherein each server sends $O(\Delta \cdot d^{\max(2,p/2+1)} \cdot \mathrm{polylog}(|V|))$ rows to its neighbors in each round and at the end of the protocol, with probability $\geq 9/10$, each server v computes an ℓ_p subspace embedding for the matrix $A^{(\Delta,v)}$.*

In each of the rounds, the information sent by a server to all its neighbors is the same thus minimizing the amount of computation to be performed and also allows for efficient communication over multicast networks.

We use the above result to solve ℓ_p linear regression and Frobenius norm low rank approximation problems in the personalized CONGEST model.

1.6 Bibliographic Remarks

Most of the work in this thesis has already been published or has been accepted for publication.

- **Part I : The Classic Setting**
    - Chapter 3 is based on [CCKW22]
    - Chapter 4 is based on [FKW21]
    - Chapter 5 is based on [KW23]
    - Chapter 6 is based on [KW21]
    - Chapter 7 is based on [KW20]
    - Chapter 8 is based on [KW22]
    - Chapter 9 is based on [KMZ23]
- **Part II : The Streaming Setting**
    - Chapter 10 is based on [KPTW23]
    - Chapter 11 is based on [EKM$^+$24b]
- **Part III : The Distributed Setting**
    - Chapters 13 and 14 are based on [EKM$^+$24a]

Chapter 2

Preliminaries

2.1 Notation

Given an integer $n \geq 1$, we use $[n]$ to denote the set $\{1, \dots, n\}$. We use small letters such as a, b, x, y, z to denote vectors and capital letters such as A, B, X, Y, Z to denote matrices. We use bold symbols such as $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ and $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ to denote random variables.

Given $x, y, a, b \geq 0$, we use the notation $x = (a, b)y$ to denote $ay \leq x \leq by$. When $a = 1 - \varepsilon$ and $b = 1 + \varepsilon$, we abbreviate the notation to $x = (1 \pm \varepsilon)y$.

Dimensions and Indexing. All the matrices in this thesis have size $n \times d$ unless specified otherwise and all the vectors are n-dimensional unless specified otherwise. Given an $n \times d$ matrix A and index i, we use A_{i*} to denote the d-dimensional vector represented by the i-th row of matrix A and A_{*i} to denote the n-dimensional vector represented by the i-th column of matrix A. Given two indices i and j, we use $A_{i,j}$ to denote the entry in the i-th row and j-th column of the matrix A. For a vector x and an index i, we use x_i to denote the i-th coordinate of the vector x.

Asymptotics. We use standard notations $O(\cdot), \Omega(\cdot), \Theta(\cdot)$ to denote the asymptotic behavior of the functions. We use $\widetilde{O}(f(n))$ to denote the set of functions $O(f(n) \cdot \operatorname{polylog} n)$, and more generally to suppress the multiplicative terms that are polynomial in the logarithms of parameters of interest. Similarly we use $\widetilde{\Omega}(f(n))$ to denote the set of functions $\Omega(f(n)/\operatorname{polylog}(n))$ and $\widetilde{\Theta}(f(n))$ to denote the functions in $\widetilde{O}(f(n)) \cap \widetilde{\Omega}(f(n))$.

Norms and SVD. For an arbitrary vector x and $p \geq 1$, we use $\|x\|_p$ to denote the ℓ_p norm of x defined as $(\sum_{i=1}^{n} |x_i|^p)^{1/p}$. For an arbitrary A, we use $\|A\|_{\mathrm{F}}$ to denote the Frobenius norm defined as $(\sum_{i,j} A_{i,j}^2)^{1/2}$ and $\|A\|_2$ to denote the operator norm defined as $\max_{x \neq 0} \|Ax\|_2/\|x\|_2$.

Given an arbitrary matrix A, we typically use the matrices $U, \Sigma, V^{\mathrm{T}}$ with appropriate dimensions so that $A = U\Sigma V^{\mathrm{T}}$ is the singular value decomposition (SVD). We use $\sigma_1(A) \geq \cdots \geq \sigma_{\min(n,d)}(A)$

to denote the singular values of A. Given $p \geq 1$, we use the notation $\|A\|_{\mathrm{S}_p}$ to denote the Schatten-p norm of A defined as $(\sum_i \sigma_i(A)^p)^{1/p}$. Note that $\|A\|_{\mathrm{S}_\infty} = \|A\|_2$ and $\|A\|_{\mathrm{S}_2} = \|A\|_{\mathrm{F}}$.

Given an $n \times d$ matrix A and $p \in [1, \infty]$, we define $\|A\|_{(p,2)}$ as the ℓ_p norm of the vector $(\|A_{1*}\|_2, \ldots, \|A_{n*}\|_2)$. Note that $\|\cdot\|_{(p,2)}$ satisfies the triangle inequality and is a norm.

Pseudoinverse and Orthogonal Projections. Given an $n \times d$ matrix A, we define A^+ to be the Moore-Penrose pseudoinverse of A. The matrix A^+ has a dimension $d \times n$ and is the unique matrix that satisfies: (i) $AA^+A = A$, (ii) $A^+AA^+ = A^+$, (iii) $(AA^+)^{\mathrm{T}} = AA^+$ and (iv) $(A^+A)^{\mathrm{T}} = A^+A$. If $A = U\Sigma V^{\mathrm{T}}$ is the singular value decomposition of the matrix A, then $A^+ = V\Sigma^+U^{\mathrm{T}}$, where Σ^+ has shape $d \times n$ and $(\Sigma^+)_{i,i} = 1/\Sigma_{i,i}$ if $\Sigma_{i,i} \neq 0$ else $\Sigma^+_{i,i} = 0$. If $\mathrm{rank}(A) = d$, then we also have $A^+ = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$.

Given a subspace V, we use $\mathbb{P}_V$ to denote the orthogonal projection matrix onto the subspace V. For all x, the vector $y = \mathbb{P}_V x$ denotes the nearest point, measured in Euclidean distance, to x in the subspace V. For a matrix A, we use $\mathbb{P}_A$ to denote $\mathbb{P}_{\mathrm{colspan}(A)}$. Given any matrix A, we have $\mathbb{P}_A = AA^+$.

Löwner Ordering. Given two symmetric matrices A and B, we define $A \preceq B$ if the matrix $B - A$ is positive semidefinite i.e., for all vectors x, $x^{\mathrm{T}}(B - A)x \geq 0$.

2.2 Subspace Embeddings

Let $V \subseteq \mathbb{R}^n$ be a d-dimensional subspace. Let S be an $m \times n$ matrix. We say that S is an ℓ_p subspace embedding for V with distortion $\alpha > 1$ if for all $x \in V$,

$$\|Sx\|_p \leq \|x\|_p \leq \alpha\|Sx\|_p.$$

The parameter m is called the *sketch dimension*. The subspace V is usually defined as the column space of a matrix A. In that case, we abuse the notation and call the matrix S as the subspace embedding of matrix A.

For $\varepsilon < 1$, we say that S is an ε ℓ_p subspace embedding for V if for all $x \in V$,

$$(1 - \varepsilon)\|Sx\|_p \leq \|x\|_p \leq (1 + \varepsilon)\|Sx\|_p.$$

2.2.1 Oblivious Subspace Embeddings

A random $m \times n$ matrix $\boldsymbol{S}$ is called an oblivious (α, δ) ℓ_p subspace embedding (OSE) if for *any* $n \times d$ matrix A, with probability $\geq 1 - \delta$, the matrix $\boldsymbol{S}$ is an ℓ_p subspace embedding for A with distortion α. An important property of a subspace embedding $\boldsymbol{S}$ is the time it takes to compute $\boldsymbol{S} \cdot A$ given a matrix A. In the literature, there are many known oblivious constructions of subspace embeddings with various trade-offs between m, α, δ and the time to compute $\boldsymbol{S} \cdot A$ for an arbitrary matrix A. We will discuss a few constructions used in this thesis below for the case of $p = 2$.

Dense Gaussian Ensemble. If each entry of $\boldsymbol{S}$ is an appropriately scaled independent standard Gaussian random variable and $m = C\varepsilon^{-2} \cdot (d + \log 1/\delta)$ for a large enough constant C, then $\boldsymbol{S}$ is an oblivious $(1+\varepsilon, \delta)$ ℓ_2 subspace embedding. Nelson and Nguyên [NN14] show that $m = \Omega(\varepsilon^{-2} \cdot (d + \log 1/\delta))$ for any oblivious $(1+\varepsilon, \delta)$ ℓ_2 subspace embedding and therefore dense Gaussian subspace embeddings have the optimal sketch dimension up to constant factors.

The main issue with dense Gaussian ensemble is that given the random matrix $\boldsymbol{S}$ and an arbitrary $n \cdot d$ matrix A, it takes $O(\min(m \cdot \text{nnz}(A), mnd))$ time to compute $\boldsymbol{S} \cdot A$ without using fast matrix multiplication algorithms. This time complexity is prohibitive, and is usually the amount of time required to solve a problem directly without using any subspace embeddings at all. Hence, we need subspace embedding constructions that can be applied quickly.

Subsampled Randomized Hadamard Transform (SRHT). SRHT is defined as the following random matrix $\boldsymbol{S} = \boldsymbol{P} \cdot H \cdot \boldsymbol{D}$, where $\boldsymbol{D}$ is an $n \times n$ diagonal matrix with each diagonal entry being ± 1 with probability $1/2$ each, H is the $n \times n$ Hadamard matrix and each row of $\boldsymbol{P}$ is an $m \times n$ matrix with each row being a uniform random vector drawn from the set $\{ e_1, \ldots, e_n \}$ where e_j denotes the n-dimensional coordinate with 1 in the j-th coordinate and 0 everywhere else. For an appropriate scaling factor γ, the random matrix $\gamma \cdot \boldsymbol{S}$ is an oblivious $(1+\varepsilon, \delta)$ ℓ_2 subspace embedding if $m = C\varepsilon^{-2} \cdot d \log^2(n/\delta)$ for a large enough constant C. While SRHT has a worse sketching dimension compared to the dense Gaussian matrix, the advantage is that given any $n \times d$ matrix A, we can compute the matrix $\boldsymbol{S} \cdot A$ in time $O(nd \log n)$ using the fast divide-and-conquer algorithm for multiplying a matrix with a Hadamard matrix.

This significantly improves upon the time required for multiplying a matrix A with a Gaussian subspace embedding. A drawback of SRHT is that it does not utilize the sparsity of the matrix A, as in, even when $\text{nnz}(A) = O(n)$, it takes $O(nd \log n)$ time to compute the matrix $\boldsymbol{S} \cdot A$.

CountSketch. The $m \times n$ CountSketch matrix $\boldsymbol{S}$ is defined as follows: each column of the matrix $\boldsymbol{S}$ has exactly one nonzero entry at a uniform random location and the nonzero entry is equal to ± 1 with $1/2$ probability each. Note that given a matrix A, we can compute $\boldsymbol{S} \cdot A$ in time $O(\text{nnz}(A))$ and therefore we can very quickly apply the CountSketch matrix to an arbitrary matrix A. Clarkson and Woodruff [CW15] showed that if $m = \text{poly}(d/\varepsilon\delta)$, then $\boldsymbol{S}$ is an oblivious $(1+\varepsilon, \delta)$ ℓ_2 subspace embedding. Nelson and Nguyên showed that $m = Cd^2/\varepsilon^2\delta$ suffices. Compared to dense Gaussians and SRHT, the CountSketch OSE requires a much larger sketch dimension and has a worse dependence on the failure probability δ.

Oblivious Subspace Norm-Approximating Projections (OSNAP). Generalizing the CountSketch construction, Nelson and Nguyên [NN13] construct a random matrix $\boldsymbol{S}$. Given $\gamma > 0$, they construct a $(1+\varepsilon, \delta)$ oblivious ℓ_2 subspace embedding matrix $\boldsymbol{S}$ with the number of rows $m \ge Cd^{1+\gamma}/\varepsilon^2\delta^\gamma$. Given a matrix A, their construction can be applied in time $\widetilde{O}_\gamma(\text{nnz}(A)/\varepsilon)$[1]. Cohen [Coh16] improved the analysis and gave a construction with $m = O(d^{1+\gamma}/\varepsilon^2\delta^\gamma)$ that can be applied to a matrix A in time $O(\text{nnz}(A)/\gamma\varepsilon)$.

2.2.2 Non-oblivious Subspace Embeddings

We will now describe sampling based constructions to obtain subspace embeddings. We first define ℓ_p leverage scores.

Leverage Scores. Given an $n \times d$ matrix A with rows $a_1, \dots, a_n \in \mathbb{R}^d$, and $p \ge 1$, the ℓ_p leverage score of the i-th row is defined as

$$\tau_i^{\ell_p}(A) = \max_{x : Ax \neq 0} \frac{|\langle a_i, x\rangle|^p}{\|Ax\|_p^p}.$$

Lemma 2.2.1 ([MMWY22, Lemma 2.6])**.** *For any $n \times d$ matrix and $p \ge 1$, $\sum_{i=1}^n \tau_i^{\ell_p}(A) \le d^{\max(1,p/2)}$.*

Using standard concentration inequalities and a net argument, we can show that leverage scores can be used to sample a matrix that is an ℓ_p subspace embedding for the column space of A with a large probability. We obtain the following theorem.

Theorem 2.2.2. *Given an $n \times d$ matrix A and $p \ge 1$, let $q \in [0, 1]^n$ be such that for all $i \in [n]$,*

$$q_i \ge C\varepsilon^{-2} \cdot \tau_i^{\ell_p}(A)(d \log(d/\varepsilon) + \log 1/\delta).$$

Let $\boldsymbol{S}$ be an $n \times n$ random diagonal matrix defined as follows: with probability $\rho_i \coloneqq \min(1, q_i)$, set $\boldsymbol{S}_{ii} = 1/\rho_i^{1/p}$, else set $\boldsymbol{S}_{ii} = 0$. With probability $\ge 1 - \delta$, $\frac{\boldsymbol{S}}{1-\varepsilon/2}$ is an ℓ_p subspace embedding for A with distortion $1 + \varepsilon$.

With a large probability, the matrix $\boldsymbol{S}$ has $O(\sum_i \min(1, q_i))$ nonzero entries. Thus, with accurate ℓ_p sensitivity estimates, we can construct subspace embeddings with a sketching dimension of $m = \widetilde{O}(d^{\max(2,p/2+1)})$. For $p = 2$, using the matrix Chernoff bounds, we can show that setting $q_i \ge C\varepsilon^{-2}\tau_i^{\ell_2}(A)(\log(d/\varepsilon\delta))$ suffices. Hence, accurate ℓ_2 sensitivities can be used to compute a subspace embedding with $\widetilde{O}(d)$ rows.

While for $p = 2$, sensitivities let us obtain subspace embeddings of near-optimal sketch sizes, it turns out sensitivities are sub-optimal for other values of p. We will now define the so-called Lewis weights and show that they can be used to obtain better subspace embeddings for $p \neq 2$ than those obtained by using sensitivities.

[1]The notation hides multiplicative factors in $\text{poly}(1/\gamma)$.

p	δ	N
$p = 1$	$1/\text{poly}(d)$	$d \log(d/\varepsilon)/\varepsilon^2$
$p = 1$	$\Omega(1)$	$d \log d/\varepsilon^2$
$p \in (1, 2)$	$\Omega(1)$	$d \log(d/\varepsilon) \log^2(\log d/\varepsilon)/\varepsilon^2$
$p > 2$	$1/\text{poly}(d)$	$d^{p/2} \log(d) \log(1/\varepsilon)/\varepsilon^5$

Table 2.1: Valid values for N

Lewis Weights. Given an $n \times d$ matrix A, Lewis weights are the unique weights $w_1, \ldots, w_n \geq 0$ that satisfy the property

$$w_i = \tau_i^{\ell_2}(W^{1/2-1/p}A)$$

where $W = \text{diag}(w_1, \ldots, w_n)$. At the outset, it is not clear if such weights must even exist due to the circular definition. Lewis [Lew78] showed the existence and uniqueness of such weights. Now, since the ℓ_2 leverage scores of any $n \times d$ matrix sum to at most d, we note that the Lewis weights sum up to at most d as well.

Theorem 2.2.3 ([CP15, Theorem 7.1]). *Given an $n \times d$ matrix A with Lewis weights $w_1, \ldots, w_n \geq 0$, let q_i be any set of sampling weights satisfying $\sum_i q_i = N$,*

$$q_i \geq f(d, N, p, \varepsilon, \delta) \cdot w_i.$$

Let S be a sketching matrix N rows where each row is independently chosen as the i-th standard basis vector times $1/q_i^{1/p}$ with probability q_i/N, then with probability $\geq 1 - \delta$, $\frac{S}{1-\varepsilon}$ is an ℓ_p subspace embedding for A with a distortion at most $1 + \varepsilon$. Valid values for N for various parameter settings is given in Table 2.1.

Part I

The Classic Setting

Chapter 3

Near-Optimal Algorithms for Linear Algebra in the Current Matrix Multiplication Time

3.1 Introduction

We obtain several new results for fundamental problems in numerical linear algebra, in many cases removing, in particular, the *last* log factor to obtain a running time that is truly linear in the input sparsity, and with lower-order terms that are close to optimal. We note that the bottleneck in improving prior work, including such removal of last logarithmic factors, involved well-known conjectures to construct *Sparse Johnson-Lindenstrauss transforms* (see Conjecture 14 in [NN13]).

To sidestep this conjecture, our key idea is to show that composing a sparse random sign matrix with an appropriate flattening transform based on explicit embeddings of ℓ_2 into ℓ_1 [Ind07] gives an OSE. Together with OSNAP embeddings [NN13, Coh16], we obtain the first oblivious subspace embedding for k-dimensional subspaces that has $o(k\log(k))$ rows and that can be applied to a matrix A in time asymptotically less than both $\mathrm{nnz}(A)\log k$ and $k^{\omega}\log k$, where $\mathrm{nnz}(A)$ is the number of nonzero entries in the matrix A, and $\omega \approx 2.37$ is the exponent of fast matrix multiplication [WXXZ24]. This scheme removes a log factor that has thus far remained both a nuisance and an impediment to optimal algorithms. Our main embedding result is as follows:

Theorem 3.1.1 (Fast Subspace Embedding, informal Theorem 3.4.3)**.** *Given an $n \times k$ matrix, there is a distribution $\mathcal{S}$ over matrices with $k\,\mathrm{poly}(\log\log k)$ rows such that, for $\mathbf{S} \sim \mathcal{S}$, with probability $\geq 99/100$, for all vectors $x \in \mathbb{R}^k$*

$$\|Ax\|_2 \leq \|SAx\|_2 \leq \exp(\mathrm{poly}(\log\log k))\|Ax\|_2.$$

For $\mathbf{S} \sim \mathcal{S}$, with probability $\geq 95/100$, the matrix $\mathbf{S}A$ can be computed in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + k^{2+\gamma+o(1)})$ for any constant $\gamma > 0$.

Using our subspace embedding, together with additional ideas, we obtain nearly optimal (up to $\log\log$ factors in the sub-linear terms) running times for fundamental problems in classical linear algebra including computing matrix rank, finding a set of linearly independent rows, and linear regression. Further, for regression and low-rank approximation, we obtain the first optimal algorithms for the current matrix multiplication exponent. We begin with least-squares regression:

Theorem 3.1.2 (Least-Squares Regression, informal Theorem 3.5.5)**.** *Given a full rank $n \times k$ matrix A, $k \leq n$, and vector b, there exists an algorithm that computes $\hat{x}$ such that $\|A\hat{x}-b\|_2 \leq (1+\varepsilon)\min_x \|Ax-b\|_2$ in time*

$$O\left(\frac{\operatorname{nnz}(A)}{\gamma} + k^{\omega}\operatorname{poly}(\log\log(k)) + \frac{1}{\operatorname{poly}(\varepsilon)}k^{2+o(1)}n^{\gamma+o(1)}\right)$$

for any constant $\gamma > 0$ small enough.

For constant ε and $k = n^{\Omega(1)}$, the running time obtained is within a $\operatorname{poly}(\log\log(n))$ factor of optimal, for the current matrix multiplication constant. Further, it improves on prior work [CW17, MM13, NN13, BDN15, CLM⁺15, CNW15] describing algorithms with an additional $\log(n)$ factor multiplying either the leading $\operatorname{nnz}(A)$ term, or that is $\operatorname{nnz}(A)$ time but has a $k^{\omega}\log k$ additive term or worse. We note that our additive term is only $k^{\omega}\operatorname{poly}(\log\log k)$, for the current matrix multiplication exponent ω, when $k = n^{\Omega(1)}$. Importantly, up to a $\operatorname{poly}(\log\log k)$ factor, our bound is best possible, and thus we remove the last logarithmic factor even in the additive term. As we explain more below, the issue with previous work is that to obtain a sketching dimension of $O(k)$, for constant ε, one needs either $\operatorname{nnz}(A)k$ time to directly perform a multiplication with a dense Sub-Gaussian matrix, or at least $k^{\omega}\log k$ time to compose a dense Sub-Gaussian sketch with a sparse sketch. We avoid this using our new subspace embedding, given by Theorem 3.4.3.

We note that simply sketching on the left with a CountSketch matrix and solving the sketched problem attains an optimal $O(\operatorname{nnz}(A))$ running time for $k = O(n^c)$ for a sufficiently small constant $c > 0$, and so our theorems are most interesting when $k = \Omega(n^c)$.

Next, we show a similar result holds for low-rank approximation (LRA):

Theorem 3.1.3 (LRA in Current Matrix Multiplication Time, informal Theorem 3.5.13)**.** *Given $\varepsilon > 0$, an $n \times d$ matrix A and $k \leq \min(n, d)$, $k = \max(n, d)^{\Omega(1)}$, there exists an algorithm that runs in*

$$O\left(\operatorname{nnz}(A) + \frac{(n+d)k^{\omega-1}}{\varepsilon} + \frac{(n+d)k^{1.01}}{\varepsilon} + \operatorname{poly}(\varepsilon^{-1}k)\right)$$

time and outputs two matrices $V \in \mathbb{R}^{n\times k}$ and $\widetilde{X} \in \mathbb{R}^{k\times d}$, with $V^{\mathrm{T}}V = I_k$, such that

$$\|A - V \cdot \widetilde{X}\|_{\mathrm{F}} \leq (1+\varepsilon)\|A - [A]_k\|_{\mathrm{F}}.$$

For the current matrix multiplication exponent, the running time is $O(\operatorname{nnz}(A) + (n+d)k^{\omega-1})$ for constant ε. In contrast, existing low rank approximation algorithms [CW17, MM13, NN13, BDN15, CEM⁺15, CLM⁺15, CNW15, CMM17] take time at least $\operatorname{nnz}(A)\log n$ or $dk^{\omega-1}\log k$ or worse. Thus, as with least squares regression, we remove the last logarithmic factor in both the $\operatorname{nnz}(A)$ term and the leading additive term.

We also give construction of a non-oblivious $1+\varepsilon$ subspace embeddings with $O(k\log(k)/\varepsilon^2)$ rows that have better running times than earlier subspace embeddings with $O(k\log(k)/\varepsilon^2)$ rows, such as approximate leverage score sampling and OSNAP embeddings.

Theorem 3.1.4 (Subspace Embeddings, informal Theorem 3.5.4)**.** *Given a matrix $A \in \mathbb{R}^{n\times k}$, there is a* non-oblivious *subspace embedding S with $O(k\log(k)/\varepsilon^2)$ rows that can be computed and applied to the matrix A in time $O(\mathrm{nnz}(A) + k^{\omega}\,\mathrm{poly}(\log\log k) + \mathrm{poly}(\varepsilon^{-1})k^{2.1+o(1)})$ for $k = n^{\Omega(1)}$.*

Finally, we obtain faster algorithms for computing the rank of a matrix and finding a full-rank set of rows.

Theorem 3.1.5 (Matrix Rank and Finding a Basis, informal Theorem 3.5.9 and 3.5.12)**.** *Given an $n\times d$ matrix A, there exists a randomized algorithm to compute $k = \mathrm{rank}(A)$ in $O(\mathrm{nnz}(A) + k^{\omega})$ time, where ω is the matrix multiplication constant. Further, the algorithm can find a set of k linearly independent rows in $O(\mathrm{nnz}(A) + k^{\omega}\log\log(n))$ time.*

We note that this result improves prior work by Cheung, Kwok and Lau [CKL13], in the case of matrices with real numbers, who obtain an $O(\mathrm{nnz}(A)\log(k) + k^{\omega})$ time algorithm to compute matrix rank and an $O(\log(n)(\mathrm{nnz}(A) + k^{\omega}))$ time algorithm to find a full-rank set of rows.

The following table lists our running times for $k \le n$ and $k = n^{\Omega(1)}$, assuming $\omega > 2$, and putting some terms to constant values (such as 2.1 instead of $2+\gamma$). See theorem statements for exact running times.

Application	Running time (up to constant factors)
ε Subspace Embeddings	$\mathrm{nnz}(A) + \varepsilon^{-3}k^{2.1+o(1)} + k^{\omega}\,\mathrm{poly}(\log\log(k))$
ε approximate linear regression	$\mathrm{nnz}(A) + \varepsilon^{-3}k^{2.1+o(1)} + k^{\omega}\,\mathrm{poly}(\log\log(k))$
Linearly Independent Rows	$\mathrm{nnz}(A) + k^{\omega}\,\mathrm{poly}(\log\log(k)) + k^{2+o(1)}$
0.01 Low-Rank Approximation	$\mathrm{nnz}(A) + (n+d)k^{\omega-1}$

3.2 Technical Overview

Before this work, the only known oblivious subspace embedding for a k dimensional subspace with $o(k\log(k))$ rows is a dense matrix of $O(k)$ rows with independent sub-Gaussian random variables. This embedding can be applied to a matrix A in time $\Omega(\mathrm{nnz}(A)\cdot k)$. All other subspace embedding constructions that are faster to apply have at least $\Omega(k\log(k))$ rows. Obtaining a subspace embedding with few rows is important to speed up the further downstream tasks such as finding a maximal set of linearly independent rows of a matrix, computing approximate leverage scores, low rank approximation, etc.

We analyze the properties required of a k-dimensional subspace $V \subseteq \mathbb{R}^n$, $n = \widetilde{O}(k)$, such that a sparse random sign matrix with $o(k\log(k))$ rows can be a subspace embedding for V. The advantage of the sparsity is that the embedding can be applied to a vector quickly. Suppose every unit vector in the subspace V has at least a constant c fraction of coordinates that have a magnitude of at least

$\widetilde{\Omega}(1/\sqrt{k})$. Let x be an arbitrary unit vector in the subspace V. Now consider a random matrix $\boldsymbol{G}$ where each entry is either 0 with probability $1-p$ and ± 1 with probability $p/2$ each. For $p = \Theta(1/n)$, as at least a constant c fraction of the coordinates of the vector x have a magnitude $\widetilde{\Omega}(1/\sqrt{k})$, each row of the matrix $\boldsymbol{G}$ has $\Omega(1)$ probability of hitting one of the large coordinates of the vector x. Conditioned on a row $\boldsymbol{G}_{i*}$ hitting one of the large coordinates of x, we have $|\boldsymbol{G}_{i*}x| \ge \widetilde{\Omega}(1/\sqrt{k})$ with probability $\ge 1/2$ by using the random signs. Thus, with at least a constant probability, for a row $\boldsymbol{G}_{i*}$, $|\boldsymbol{G}_{i*}x|^2 \ge \widetilde{\Omega}(1/k)$. If the matrix $\boldsymbol{G}$ has $\Omega(k)$ rows, using the Chernoff bound, we have that with very high probability, $\|\boldsymbol{G}x\|_2^2 \ge \widetilde{\Omega}(1)$, which suffices to union bound over a suitable net of unit vectors in a k-dimensional subspace. On the other hand, showing that $\|\boldsymbol{G}\|_2$ is small and that it does not increase the norm of any unit vector by a lot is much easier. For the probability p that we consider, each row and column of the matrix $\boldsymbol{G}$ only has $\widetilde{O}(1)$ nonzero entries with high probability. As all the nonzero entries are at either ± 1, we can bound the operator norm $\|\boldsymbol{G}\|_2$ by $O(1)$. This implies that for any unit vector x, $\|\boldsymbol{G}x\|_2^2 \le \widetilde{O}(1)$.

The above argument shows that if a subspace has the property that every unit vector in the subspace has a *large* number of *large* coordinates, then a random sparse sign matrix is a subspace embedding with small distortion for that subspace. We call subspaces having this property *flat*. But of course, the column space of the matrix to which we want to apply the embedding may not have this property. Let $V_1 \subseteq \mathbb{R}^n$ be the column space of the given matrix A. If we can find a linear map $\mathcal{F}$ that maps vectors in the subspace V_1 to a *flat* subspace V_2 and if $\mathcal{F}$ preserves the Euclidean norms of the vectors, then we have that $\|\boldsymbol{G}\mathcal{F}x\|_2 \approx \|\mathcal{F}x\|_2 \approx \|x\|_2$ for all vectors $x \in V_1$. As we show later, by paying some cost in running time, we can assume that $n = O(k\log(k))$ by first applying a series of suitable **OSNAP** embeddings. To obtain such a mapping $\mathcal{F}$, we use the $\ell_2 \to \ell_1$ embedding F of [Ind07]. We show that recursively applying the linear map F gives a linear map $\mathcal{F} : n \to n^{1+o(1)}$ with the property that for all unit vectors x, $\|\mathcal{F}x\|_2 \approx 1$ and $\|\mathcal{F}x\|_1 \ge \widetilde{\Omega}(\sqrt{n})$. This property immediately shows that the vector $\mathcal{F}x$ must have a large number of large coordinates and therefore that the subspace range($\mathcal{F}$) is *flat*. We only obtain that a $1/n^{o(1)}$ fraction of the coordinates are large but it is sufficient for our purposes. We also show that the sequence of **OSNAP**, the mapping of [Ind07] which we call Indyk, and the sparse random sign embeddings can be applied to a matrix $A \in \mathbb{R}^{n\times k}$ in time $O(\gamma^{-1}\operatorname{nnz}(A) + k^{2+\gamma+o(1)})$ for any constant $\gamma > 0$.

$1+\varepsilon$ Subspace Embeddings. We use our $\exp(\operatorname{poly}(\log\log k))$ distortion subspace embedding construction to obtain $1+\varepsilon$ *non-oblivious* subspace embeddings using approximate leverage scores obtained by using a preconditioner. Let $A \in \mathbb{R}^{n\times k}$. Earlier algorithms to compute approximate leverage scores can be described as follows : (i) Compute $\boldsymbol{S}A$ where $\boldsymbol{S}$ is a subspace embedding for the column space of A, (ii) Compute an orthonormal matrix Q and matrix R^{-1} such that $\boldsymbol{S}A = QR^{-1}$, and (iii) Compute the approximate leverage scores $\widetilde{\ell_i^2} = \|A_iR\|_2^2$.

Thus, to make computing approximate leverage scores faster, we need a subspace embedding $\boldsymbol{S}$ that can be quickly applied to matrix A to make step (i) faster while also having a fewer number of rows to make the computation of the QR-decomposition in step (ii) faster. As discussed, our subspace embedding construction $\boldsymbol{S}$ has both of these desired properties. In step (iii), instead of computing $\|A_{i*}R\|_2^2$ exactly, a Gaussian matrix $\boldsymbol{G}$ with $O(\log(n))$ columns is used so that for all the rows $i \in [n]$, $\|A_{i*}R\boldsymbol{G}\|_2^2 \approx \|A_{i*}R\|_2^2$, which is a standard idea [DMMW12]. However, computing the matrix $AR\boldsymbol{G}$ takes $\Omega(\text{nnz}(A)\log(n))$ time. We consider using a Gaussian matrix with only $O(1/\gamma)$ columns for an absolute constant $\gamma > 0$, which is also a standard idea in this area. Consider an arbitrary vector v and let $\boldsymbol{g}$ be a vector of i.i.d. normal random variables. Then we have the probability that $|\langle v, \boldsymbol{g}\rangle| \le \|v\|_2/n^{\gamma}$ is at most $1/n^{\gamma}$. If $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_t$ are independent Gaussian vectors for $t = O(1/\gamma)$, then at least one of the values $|\langle v, \boldsymbol{g}_i\rangle|$ is at least $\|v\|_2/n^{\gamma}$ with probability $\ge 1 - 1/n^2$. If $\boldsymbol{G}$ is a matrix with $\boldsymbol{g}_j$ as its columns, we therefore have that $\|A_{i*}R\boldsymbol{G}\|_2^2 \ge \|A_{i*}R\|_2^2/n^{2\gamma}$ for all i. We also argue that $\|A_{i*}R\boldsymbol{G}\|_2^2 = O(\|A_{i*}R\|_2^2\log(n))$ for all $i \in [n]$. Now the matrix $AR\boldsymbol{G}$ and the approximations $\|A_{i*}R\boldsymbol{G}\|_2^2$ can be computed in time $O(\gamma^{-1}(\text{nnz}(A) + k^2))$. Therefore, we can obtain over-estimates to the leverage scores. Using over-estimates to the leverage score sampling probabilities, we first sample rows and then compute accurate leverage scores only for the rows that are sampled. Then we employ a rejection step, in which we reject rows randomly based on the probabilities computed using accurate leverage scores, and finally we show that we obtain a sample from the leverage score sampling distribution. As we compute accurate leverage scores only for the rows that are sampled in the first stage, we do not incur the $O(\text{nnz}(A)\log(n))$ factor. We then compose our leverage score embedding with an OSNAP embedding to obtain a $1+\varepsilon$ embedding with $O(k\log(k)/\varepsilon^2)$ rows, which is faster than previous constructions.

Computing Linearly Independent Rows. We give an algorithm to compute a maximal set of linearly independent rows of a matrix $A \in \mathbb{R}^{n\times d}$ of rank k in time $O(\text{nnz}(A) + k^{\omega}\,\text{poly}(\log\log(n)))$. Using the rank-preserving sketches of [CKL13], we can assume without loss of generality that $d = ck$ for a constant c. The crucial idea here is that a leverage score sample of the matrix A, with high probability, must contain a set of k linearly independent rows. Therefore, directly applying the above leverage score sampling algorithm for constant ε gives, in time $O(\gamma^{-1}\,\text{nnz}(A) + n^{\gamma}k^{2+o(1)} + k^{\omega}\,\text{poly}(\log\log(n)))$, for any constant γ, a set of $O(k\exp(\text{poly}(\log\log k)))$ rows of the matrix A that must contain a set of k linearly independent rows. To obtain a running time that does not depend on n^{γ}, we show that instead of running leverage score sampling on the matrix A, we can apply reductions as in [CKL13] to reduce the problem to computing linearly independent rows of a submatrix A' with $\text{nnz}(A') \le \min(\text{nnz}(A)/\text{poly}(\log(n)), O(k^2))$ and with $n/\text{poly}(\log(n))$ rows. This reduction can be performed in time $O(\text{nnz}(A) + k^{\omega}\log\log(n))$. After this reduction, we perform leverage score sampling for the matrix A' as described above with constant ε and $\gamma = O(1/\log(n))$ to obtain a matrix $\boldsymbol{S}_{\text{lev}}$ that selects and scales $O(k\exp(\text{poly}(\log\log k)))$ rows randomly according to the leverage score distribution such that for all x, $\|\boldsymbol{S}_{\text{lev}}A'x\|_2 = (1 \pm 1/2)\|A'x\|_2$. In particular, the guarantee implies that $\text{rowspace}(\boldsymbol{S}_{\text{lev}}A') = \text{rowspace}(A')$. Therefore there are k linearly independent rows among the $O(k\exp(\text{poly}(\log\log k)))$ rows sampled by $\boldsymbol{S}_{\text{lev}}$. Now we can again apply the row reduction procedure mentioned above to the matrix $S_{\text{lev}}A'$, to finally obtain, in time $O(k^{2+o(1)} + k^{\omega}\operatorname{poly}(\log\log k))$, a set of $O(k)$ rows that, with high probability, contain a set of k linearly independent rows. These rows can now be identified in time $O(k^{\omega})$. Thus, we obtain that in time $O(\operatorname{nnz}(A) + k^{\omega}\operatorname{poly}(\log\log n) + k^{2+o(1)})$, we can compute a set of k linearly independent rows of a rank k matrix A. The leverage score subspace embedding having $k\operatorname{poly}(\log\log k)$ rows turns out to be crucial to obtain a running time that depends on $k^{\omega}\operatorname{poly}(\log\log n)$ instead of the $k^{\omega}\log(n)$ dependence of earlier algorithms.

Low Rank Approximation. Finally, we give an algorithm to compute a $(1+\varepsilon)$-approximate rank-k approximation to an arbitrary matrix A. We do not need to utilize our subspace embedding construction in this algorithm, though we include it as it is also a fundamental problem in linear algebra for which we remove the last logarithmic factor. We compute a low rank approximation in two stages: (i) we first find a rank k orthonormal matrix V whose columns span a $1+\varepsilon$ approximation. (ii) we then find a right factor $\widetilde{X}$ such that $V \cdot \widetilde{X}$ is a $(1+\varepsilon)$ rank-k approximation. We obtain the left factor V by using projection-cost preserving sketches and subspace embeddings along with the CUR decomposition algorithm from [BW17], to first obtain an $O(k)$-dimensional subspace that spans an $O(1)$-approximate rank-k low rank approximation. We then perform the residual sampling algorithm of [DRVW06] to obtain a set of $O(k/\varepsilon)$ columns of the matrix A, which along with the $O(k)$ dimensional subspace we already found, span a $(1+\varepsilon)$-approximation. We then use affine embeddings to compute a left factor V that spans a $(1+\varepsilon)$-approximation.

After finding a left factor V, the matrix $V^{\mathrm{T}}A$ is the optimal right factor but it takes $\Omega(\operatorname{nnz}(A)\cdot k)$ time to compute this matrix. To avoid this, we run the CUR decomposition algorithm of Boutsidis and Woodruff [BW17] using the matrix V we found to obtain a right factor $\widetilde{X}$ such that $\|V\cdot\widetilde{X} - A\|_{\mathrm{F}} \le (1+\varepsilon)\|A - [A]_k\|_{\mathrm{F}}$.

3.3 Flattening the vectors

In this section, we argue that there is a linear mapping $\mathscr{F} : \mathbb{R}^n \to \mathbb{R}^{n^{1+o(1)}}$ such that for any unit vector $x \in \mathbb{R}^n$, the set

$$\operatorname{Large}(\mathscr{F}x) \coloneqq \{i \in [n^{1+o(1)}] \mid |(\mathscr{F}x)_i| \ge \frac{1}{\sqrt{n}\cdot\exp(\operatorname{poly}(\log\log n))}\}$$

has size $|\operatorname{Large}(\mathscr{F}x)| = \widetilde{\Omega}(n)$. In the following it will be helpful to have an abbreviation.

Definition 3.3.1. Let $\operatorname{epll}(n)$ denote the class of functions in $O(\exp(\operatorname{poly}(\log\log(n))))$.

We show that an explicit $\ell_2 \to \ell_1$ linear embedding construction of Indyk [Ind07] can be used to obtain such a mapping $\mathscr{F}$. First we define (ε, l) extractors as follows.

Definition 3.3.2 ((ε, l) extractors)**.** A bipartite graph $G = (A, B, E)$, $A = [a]$ and $B = [b]$, with each left node having degree d is an (ε, l) extractor if it has the following property. Let $\mathscr{P}$ be *any* distribution over the set A such that for all $i \in [a]$, $\mathbf{Pr}_{\mathscr{P}}[i] \leq 1/l$. Consider the distribution over B generated by the following process:

1. Sample $i \in A$ from distribution $\mathscr{P}$
2. Sample $t \in [d]$ uniformly at random and set $j = \Gamma_G(i)_t$. Here $\Gamma_G(i)$ is the ordered set of neighbors of i in the graph G and $\Gamma_G(i)_t$ is the t-th neighbor in the ordered set.

Let $G(\mathscr{P})$ be the distribution of the element j sampled by the above process. The graph G is an (ε, l) extractor if $(1/2)\sum_{j\in B} |\mathbf{Pr}_{G(\mathscr{P})}[j] - 1/b| \leq \varepsilon$. We stress that this property must hold for every distribution $\mathscr{P}$ with $\mathbf{Pr}_{\mathscr{P}}[i] \leq 1/l$ for all i.

See [Ind07] and references therein for several examples of explicit constructions of extractors. Indyk uses an extractor from [Zuc97] with the following parameters: Fix a $\delta = \Omega(1/\sqrt{n})$ and let $L = O(1/\delta^2)$ and $s = \sqrt{n}$. Let G be an (ε, l) extractor with $A = [Ln]$, $B = [b]$ for $b = n^{1/2-\kappa}$, $\kappa > 0$, $l = (1-\delta)^2 s/L$, left degree $d = (\log a)^{O(1)} = (\log Ln)^{O(1)}$ and max right degree $\Delta = O(nLd/b)$.

Theorem 3.3.3 (Theorem 1.1 of [Ind07])**.** *For any constants $\zeta, \kappa > 0$, there is an explicit linear mapping $F : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, $m = O(nLd) = n\log^{O(1)}(n)/\zeta^{O(1)}$ and a partitioning of the coordinate set $[m]$ into sets $B_1, \dots, B_b$, for $b = n^{1/2-\kappa}$, each of size at most $\Delta = n^{1/2+\kappa}\,\mathrm{epll}(n)/\zeta^{O(1)}$, such that for any $x \in \mathbb{R}^n$, $\|x\|_2 = 1$,*

$$(1 - O(\zeta))\sqrt{Ldb} \leq \sum_{j=1}^{b} \|(Fx)_{B_j}\|_2 \leq \sqrt{Ldb}.$$

Without loss of generality, we can assume that all the partitions B_j have the same size Δ by appending 0-valued coordinates, and so we have $m = n \cdot \mathrm{epll}(n)/\zeta^{O(1)}$.

We now prove the following lemma which essentially shows that an application of Indyk's embedding to a unit vector shrinks the Euclidean norm by a lot, while keeping the ℓ_1 norm $\Omega(1)$.

Lemma 3.3.4. *Let n be an arbitrary integer and $0 < \zeta, \kappa < c$ for a small enough constant c. There is an explicit linear mapping $F : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ for $m = n \cdot \mathrm{epll}(n)/\zeta^{O(1)}$ and a partitioning of $[m]$ into equal sized sets $B_1, \dots, B_b$ where $b = n^{1/2-\kappa}$ and each set B_j satisfies $|B_j| = \Delta = n^{1/2+\kappa}\,\mathrm{epll}(n)/\zeta^{O(1)}$, such that for any $x \in \mathbb{R}^n$, we have*

$$(1 - O(\zeta))\|x\|_2 \leq \sum_{j=1}^{b} \|(Fx)_{B_j}\|_2 \leq \|x\|_2$$

and

$$\|Fx\|_2^2 = \sum_{j=1}^{b} \|(Fx)_{B_j}\|_2^2 = \frac{1}{b}\|x\|_2^2.$$

Proof. In the proof of the above theorem, Indyk uses the (ε, l) construction specified above with $\delta = \zeta$ and $\varepsilon = \zeta^2$. Indyk also defines $(Fx)_{B_j} \coloneqq (Dx)_{\Gamma_G(j)}$ for $j \in [b]$, where D is a concatenation of certain L orthonormal matrices and $\Gamma_G(j) \subseteq A$ is the set of neighbors of $j \in B$ in the graph

G. For any vector x, we have $\|Dx\|_2^2 = L\|x\|_2^2$ and as the left degree of G is exactly equal to d, we have $\|Fx\|_2^2 = \sum_j \|(Fx)_{B_j}\|_2^2 = \sum_j \|(Dx)_{\Gamma_G(j)}\|_2^2 = d\|Dx\|_2^2 = Ld\|x\|_2^2$. Hence, the matrix $F/\sqrt{Ldb}$ satisfies that for any vector x,

$$\sum_{j=1}^{b} \|(\frac{F}{\sqrt{Ldb}}x)_{B_j}\|_2^2 = \frac{1}{b}\|x\|_2^2.$$

From the above theorem, we already have

$$(1 - O(\zeta))\|x\|_2 \leq \sum_{j=1}^{b} \|(\frac{F}{\sqrt{Ldb}}x)_{B_j}\|_2 \leq \|x\|_2.$$

Therefore, scaling the matrix F gives the proof. □

We apply the above lemma recursively to each of the partitions B_j for $\Theta(\log\log(n))$ levels to obtain the following theorem.

Theorem 3.3.5. *Given any n, there is an explicit map $\mathcal{F} : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ with $m = n \cdot \mathrm{epll}(n)$ such that for all unit vectors $x \in \mathbb{R}^n$, we have*

$$\|\mathcal{F}x\|_1 \geq \sqrt{n/\log n}$$

and

$$\|\mathcal{F}x\|_2^2 = 1.$$

Further, given any vector x, the vector $\mathcal{F}x$ can be computed in $n^{1+o(1)}$ time.

Proof. Let $N = \Theta(\log\log(n))$ and $\zeta = c$ be a small enough constant so that $(1-O(\zeta))^N \geq 1/\sqrt{\log n}$. Let $B_1, \ldots, B_{b_1}$ be the partitions of the coordinates of the range of F from the Lemma 3.3.4. We recursively apply the lemma for each of the partitions for N levels to obtain $\mathcal{F} : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ for $m = n \cdot \mathrm{epll}(n)$. Define $n_0 = n$ and let n_i be the number of entries in each of the i-th level partitions. Also, let $b_0 = 1$ and b_i be the number of partitions an $(i-1)$-th level partition is mapped into. From Lemma 3.3.4, we have

$$b_i = n_{i-1}^{1/2-\kappa}$$

and

$$n_i = n_{i-1}^{1/2+\kappa}\,\mathrm{epll}(n_{i-1}).$$

since ζ is a small constant. The following lemma lower bounds the number of partitions in the N-th level.

Lemma 3.3.6. *The total number of partitions in the N-th level is given by $B = b_0 \cdot b_1 \cdots b_N$ and*

$$B \geq n/2.$$

Proof. We have $B = b_1 \cdots b_N = (n_0 \cdots n_{N-1})^{1/2-\kappa}$. As $n_i \geq n^{(1/2+\kappa)^i}$, we have that $n_0 \cdots n_{N-1} \geq n^{\sum_{i=0}^{N-1}(1/2+\kappa)^i}$. Now, $\sum_{i=0}^{N-1}(1/2+\kappa)^i = (1-(1/2+\kappa)^N)/(1/2-\kappa)$ which implies $B \geq n^{1-(1/2+\kappa)^N}$. For $N = \Theta(\log\log(n))$, $(1/2+\kappa)^N \leq 1/\text{poly}(\log(n))$ and $B \geq n/2$. $\square$

This lemma implies that the N-th level has the partitions $\mathcal{B}_1, \ldots, \mathcal{B}_B$ of $[m]$ with $B \geq n/2$ and $|\mathcal{B}_j| = \text{epll}(n)$ such that for any unit vector x,

$$\frac{1}{\sqrt{\log n}}\|x\|_2 \leq (1-O(\zeta))^N\|x\|_2 \leq \sum_{j=1}^{B}\|(\mathcal{F}x)_{\mathcal{B}_j}\|_2 \leq \|x\|_2$$

and

$$\frac{1}{2B}\|x\|_2^2 \leq \frac{(1-O(\zeta))^N}{B}\|x\|_2^2 \leq \sum_{j=1}^{B}\|(\mathcal{F}x)_{\mathcal{B}_j}\|_2^2 \leq \frac{1}{B}\|x\|_2^2.$$

Finally, for a unit vector x,

$$\frac{1}{2} = \frac{1}{2}\|x\|_2 \leq \sum_{j=1}^{B}\|(\mathcal{F}x)_{\mathcal{B}_j}\|_2 \leq \sum_{j=1}^{B}\|(\mathcal{F}x)_{\mathcal{B}_j}\|_1 = \|\mathcal{F}x\|_1$$

and

$$\|\mathcal{F}x\|_2^2 = \sum_{j=1}^{B}\|(\mathcal{F}x)_{\mathcal{B}_j}\|_2^2 = \frac{1}{B}\|x\|_2^2.$$

By scaling the map $\mathcal{F}$ by $\sqrt{B}$, we complete the proof. $\square$

We now have the following corollary.

Corollary 3.3.7. *Given any unit vector x, at least $\widetilde{\Theta}(n)$ coordinates of the vector $\mathcal{F}x \in \mathbb{R}^m$ have an absolute value of at least $\eta = 1/(\sqrt{n} \cdot \text{epll}(n))$.*

Proof. Let m' be the number of coordinates of $\mathcal{F}x$ with an absolute value of at least η. Let $T \subseteq [m]$ be the set of indices of those coordinates. Then

$$\begin{aligned}\sqrt{n/\log n} \leq \|\mathcal{F}x\|_1 &= \sum_{i \notin T}|(\mathcal{F}x)_i| + \sum_{i \in T}|(\mathcal{F}x)_i| \\ &\leq \frac{m}{\sqrt{n}\cdot\text{epll}(n)} + \sqrt{\sum_{i\in T}(\mathcal{F}x)_i^2}\sqrt{|T|} \\ &\leq \frac{n\cdot\text{epll}(n)}{\sqrt{n}\cdot\text{epll}(n)} + \sqrt{m'}.\end{aligned}$$

Here we use the Cauchy-Schwarz inequality and the fact that $\|\mathcal{F}x\|_2^2 = 1$. For appropriate η chosen based on m, the above inequality implies that

$$\sqrt{m'} \geq \sqrt{n/2\log n} \implies m' \geq n/2\log n.$$

which shows that an $\widetilde{\Omega}(n)$ fraction of the coordinates of $\mathcal{F}x$ have an absolute value of at least η. □

Thus, applying Lemma 3.3.4 for $N = \Theta(\log\log(n))$ levels gives an n dimensional subspace of $\mathbb{R}^m$ for $m = n \cdot \text{epll}(n)$ such that for every unit vector x, the vector $\mathcal{F}x$ has a *large* number of *large* coordinates.

3.4 Fast Subspace Embeddings

Algorithm 3.1: FASTEMBEDDING

```
    Input: $A \in \mathbb{R}^{n\times k}, \gamma > 0$
    Output: A subspace embedding $SA$ with $O(k \cdot \text{epll}(k))$ rows
1   $S_1 \leftarrow \text{OSNAP}(A, \gamma)$ with $O(k^{1+\gamma+o(1)})$ rows
2   $S_2 \leftarrow \text{OSNAP}(S_1A, O(1/\log(n)))$ with $O(k\log(k))$ rows
3   $\mathcal{F} \leftarrow$ Indyk Embedding for $\mathbb{R}^{O(k\log(k))}$ for $\Theta(\log\log(k))$ levels with $r = k \cdot \text{epll}(k)$ rows
4   $m \leftarrow k \cdot \text{poly}(\log\log k), p \leftarrow \text{epll}(k)/r$
5   $\boldsymbol{G} \leftarrow m \times r$ random matrix where each entry is independently 0 with probability $1-p$, and
      $\pm 1$ with probability $p/2$ each
6   $SA \leftarrow \kappa \cdot \boldsymbol{G} \cdot \mathcal{F} \cdot \boldsymbol{S}_2 \cdot \boldsymbol{S}_1 A$ where $\kappa$ is an appropriate scaling factor
7   return $SA$
```

Let A be an arbitrary $n \times k$ matrix with $\text{nnz}(A)$ nonzero entries. We design a random matrix $\boldsymbol{S}$ with $k \cdot \text{poly}(\log\log(k))$ rows such that with probability $\geq 9/10$, for all vectors x,

$$\|x\|_2 \leq \|\boldsymbol{S}Ax\|_2 \leq \text{epll}(k)\|x\|_2.$$

The matrix $\boldsymbol{S}A$ can be computed in time $\text{nnz}(A) + k^{2.1+o(1)}$. The matrix $\boldsymbol{S}$ is constructed as a composition of various oblivious subspace embeddings.

We first apply OSNAP matrix $\boldsymbol{S}_1$ with $\mu = 0.1$ to obtain an $O(k^{1.1}\log(k)) \times k$ matrix $\boldsymbol{S}_1A$ in time $O(\text{nnz}(A))$. Now, $\text{nnz}(\boldsymbol{S}_1A) = O(k^{2.1}\log(k))$. Therefore, we can apply OSNAP $\boldsymbol{S}_2$ with $\mu = 1/\log(k)$, to obtain an $O(k\log k) \times k$ matrix $\boldsymbol{S}_2\boldsymbol{S}_1A$ in time $O(\text{nnz}(\boldsymbol{S}_1A) \cdot 1/\mu) = O(k^{2.1}\log^2(k))$. We also have with probability $\geq 98/100$ that

$$\|\boldsymbol{S}_2\boldsymbol{S}_1Ax\|_2 \in (1 \pm 3/10)\|Ax\|_2$$

for all vectors $x \in \mathbb{R}^k$. We then use the flattening transform $\mathcal{F}$ to obtain a constant subspace embedding for the matrix $\boldsymbol{S}_2 \cdot \boldsymbol{S}_1 \cdot A$ which also has the property that every unit vector in the column space of the matrix $\mathcal{F} \cdot \boldsymbol{S}_2 \cdot \boldsymbol{S}_1 \cdot A$ has a large number of large entries.

Theorem 3.4.1 (Indyk Embedding, Theorem 3.3.5 and Corollary 3.3.7)**.** *Given any n, there is an explicit linear map/matrix $\mathcal{F} \in \mathbb{R}^{m\times n}$ with $m = n \cdot \mathrm{epll}(n)$ such that for any vector $x \in \mathbb{R}^n$,*

$$\|\mathcal{F}x\|_2 = \|x\|_2$$

and for any unit vector x, at least $\widetilde{\Theta}(n)$ coordinates of the vector $\mathcal{F}x$ have an absolute value of at least $1/(\sqrt{n} \cdot \mathrm{epll}(n))$. Given a vector $x \in \mathbb{R}^n$, the explicit map $\mathcal{F}x$ can be computed in time $n^{1+o(1)}$.

Combining $\mathcal{F}, \boldsymbol{S}_2, \boldsymbol{S}_1$, we obtain that with probability $\geq 98/100$, for all vectors x,

$$\frac{7}{10}\|Ax\|_2 \leq \|\mathcal{F} \cdot \boldsymbol{S}_2 \cdot \boldsymbol{S}_1 \cdot Ax\|_2 \leq \frac{13}{10}\|Ax\|_2.$$

The matrix $\mathcal{F} \cdot \boldsymbol{S}_2 \cdot \boldsymbol{S}_1 \cdot A$ can be computed in time $\mathrm{nnz}(A) + k^{2.1+o(1)}$. As the matrix $\boldsymbol{S}_2 \cdot \boldsymbol{S}_1 \cdot A$ has $O(k \cdot \log(k))$ rows, the matrix $\mathcal{F}$ has $O(k\log(k) \cdot \mathrm{epll}(k)) = k \cdot \mathrm{epll}(k)$ rows and we also obtain that for any unit vector x in the column space of $\mathcal{F} \cdot S_2 \cdot S_1 \cdot A$, at least $\widetilde{\Theta}(k)$ coordinates have an absolute value of at least $1/(\sqrt{k\log k}\ \mathrm{epll}(k)) = 1/(\sqrt{k}\ \mathrm{epll}(k))$. The following theorem shows that a sparse sign matrix is a subspace embedding for a subspace with every unit vector in the subspace having a large number of large entries.

Theorem 3.4.2. *Let $A \in \mathbb{R}^{m\times k}$, with $m = k \cdot \mathrm{epll}(k)$, be a matrix such that for all unit vectors $x \in$ colspan(A), the set*

$$\mathit{Large}(x) := \left\{ i \in [m] \mid |x_i| \geq \eta = \frac{1}{\sqrt{k} \cdot \mathrm{epll}(k)} \right\}$$

satisfies $|\mathit{Large}(x)| \geq k/\mathrm{poly}(\log k)$. There is a distribution $\mathcal{G}$ over matrices with $M = k \cdot \mathrm{poly}(\log\log(k))$ rows such that for $\boldsymbol{G} \sim \mathcal{G}$, with probability $\geq 9/10$, for all vectors $x \in \mathbb{R}^k$,

$$\|Ax\|_2 \leq \|\boldsymbol{G}Ax\|_2 \leq \mathrm{epll}(k)\|Ax\|_2.$$

With probability $\geq 9/10$, the matrix $\boldsymbol{G}A$ can be computed in time $k^2 \cdot \mathrm{epll}(k)$.

Proof. Define the $M \times m$ random matrix $\boldsymbol{G}$ as follows:

$$\boldsymbol{G}_{ij} = \begin{cases} +1 & \text{with probability } p/2 \\ -1 & \text{with probability } p/2 \\ 0 & \text{with probability } 1-p \end{cases}$$

for some values of $M \leq m$ and p to be chosen later. The random variables $\boldsymbol{G}_{ij}$ are mutually independent. Let $\boldsymbol{X}_i$ be the number of nonzero entries in the i-th row of $\boldsymbol{G}$ and let $\boldsymbol{Y}_j$ be the number of nonzero entries in the i-th column of $\boldsymbol{G}$. By the Chernoff bound, for $\delta > 1$,

$$\mathbf{Pr}[\boldsymbol{X}_i \geq (1+\delta) \cdot mp] \leq \exp(-\delta mp/4) \quad \text{and} \quad \mathbf{Pr}[\boldsymbol{Y}_j \geq (1+\delta) \cdot Mp] \leq \exp(-\delta Mp/4).$$

Let p be such that $p|\text{Large}(x)| \geq 10$ for all x. As $|\text{Large}(x)| \geq k/\text{poly}(\log k)$, there is a value of p for which $pm \leq \text{epll}(k)$ and $p|\text{Large}(x)| \geq 10$ for all x. By a union bound, we obtain that with probability $\geq 99/100$, for all i and j, $\boldsymbol{X}_i \leq \text{epll}(k)$ and $\boldsymbol{Y}_j \leq \text{epll}(k)$. Thus, with probability $\geq 99/100$

$$\max_i \sum_j |\boldsymbol{G}_{ij}| = \max_i \boldsymbol{X}_i \leq \text{epll}(k) \text{ and } \max_j \sum_i |\boldsymbol{G}_{ij}| = \max_j \boldsymbol{Y}_j \leq \text{epll}(k).$$

We now have that $\|\boldsymbol{G}\|_2 \leq \sqrt{(\max_i \sum_j |\boldsymbol{G}_{ij}|)(\max_j \sum_i |\boldsymbol{G}_{ij}|)} \leq \text{epll}(k)$, which implies that for any vector y,

$$\|\boldsymbol{G} \cdot Ay\|_2 \leq \text{epll}(k)\|Ay\|_2.$$

Let the event that $\|\boldsymbol{G}\|_2 \leq \text{epll}(k)$ be $\mathscr{E}$.

We now show a contraction lower bound. Let x be an arbitrary unit vector in the column space of the matrix A. We say a row $\boldsymbol{G}_{i*}$ is *good* if $\boldsymbol{G}_{ij}$ is nonzero for some $j \in \text{Large}(x)$. We say $\boldsymbol{G}_{i*}$ is *bad* if it is not *good*. We have

$$\mathbf{Pr}[\boldsymbol{G}_{i*} \text{ is } \textit{bad}] = (1-p)^{|\text{Large}(x)|} \leq \exp(-p|\text{Large}(x)|) \leq \exp(-10) \leq 1/100.$$

Thus, $\mathbf{Pr}[\boldsymbol{G}_{i*} \text{ is } \textit{good}] \geq 99/100$.

We say a row $\boldsymbol{G}_{i*}$ is *large* if $|\boldsymbol{G}_{i*}x| \geq \eta$. Condition on the event that $\boldsymbol{G}_{i*}$ is *good*. Let $j \in \text{Large}(x) \cap \text{nnz}(\boldsymbol{G}_{i*}) \neq \emptyset$. Now, $\boldsymbol{G}_{i*}x = \sum_{j' \in \text{nnz}(\boldsymbol{G}_{i*})-j} \boldsymbol{G}_{ij'}x_{j'} + \boldsymbol{G}_{ij}x_j$. As entries of the matrix $\boldsymbol{G}$ are mutually independent, with probability $1/2$, $\boldsymbol{G}_{ij}x_j$ has the same sign as $\sum_{j' \in \text{nnz}(\boldsymbol{G}_{i*})-j} \boldsymbol{G}_{ij}x_j$, which implies that with probability $\geq 1/2$, $|\boldsymbol{G}_{i*}x| \geq |x_j| \geq \eta$. Thus,

$$\mathbf{Pr}[\boldsymbol{G}_{i*} \text{ is } \textit{large} \mid \boldsymbol{G}_{i*} \text{ is } \textit{good}] \geq 1/2$$

which implies that

$$\mathbf{Pr}[|\boldsymbol{G}_{i*}x| \geq \eta] = \mathbf{Pr}[\boldsymbol{G}_{i*} \text{ is } \textit{large}] \geq (1/2) \cdot (99/100) \geq 1/4.$$

Let l denote the number of *large* rows. As rows of the matrix $\boldsymbol{G}_{i*}$ are independent, *largeness* of rows is mutually independent. Thus, by the Chernoff bound,

$$\mathbf{Pr}[l \leq (1/2) \cdot M \cdot (1/4)] \leq \exp(-M/32).$$

We now condition on the event $\mathscr{E}$. We have

$$\mathbf{Pr}[l \le M/8 \mid \mathscr{E}] \le \frac{\mathbf{Pr}[l \le M/8]}{\mathbf{Pr}[\mathscr{E}]} \le 2\exp(-M/32).$$

Therefore, conditioned on the event $\mathscr{E}$, with probability $\ge 1 - 2\exp(-M/32)$, we have $l \ge M/8$ which implies that

$$\|\boldsymbol{G}x\|_2^2 \ge \sum_{\text{large } i} |\boldsymbol{G}_{i*}x|^2 \ge l\eta^2 \ge \frac{l}{k\ \text{epll}(k)} \ge \frac{M}{8k\ \text{epll}(k)}.$$

In what follows, we condition on the event $\mathscr{E}$. For $M = k \cdot \text{poly}(\log\log(k))$, we obtain that for a unit vector x, with probability $\ge 1 - \exp(-k\,\text{poly}(\log\log(k)))$,

$$\|\boldsymbol{G}x\|_2^2 \ge \frac{\text{poly}(\log\log(k))}{\text{epll}(k)}.$$

By suitably scaling $\boldsymbol{G}$, we obtain that for all vectors x,

$$\|\boldsymbol{G}x\|_2 \le \text{epll}(k)\|x\|_2$$

and for any unit vector x, with probability $\ge 1 - \exp(-k \cdot \text{poly}(\log\log(k)))$,

$$\|\boldsymbol{G}x\|_2 \ge 2.$$

The column space of the matrix A has dimension at most k. Let $\mathcal{N}$ be a net of the unit vectors in the column space of A such that for any $y \in \text{colspace}(A)$, $\|y\|_2 = 1$, there is an $x_y \in \mathcal{N}$, $\|x_y\|_2 = 1$ such that

$$\|x_y - y\|_2 \le \frac{1}{\|\boldsymbol{G}\|_2}.$$

As $\|\boldsymbol{G}\|_2 \le \text{epll}(k)$, there exists a net $\mathcal{N}$ of size $\exp(k \cdot \text{poly}(\log\log(k)))$. We union bound over all the net vectors to obtain that with probability $\ge 99/100$, for all net vectors $x \in \mathcal{N}$,

$$\|\boldsymbol{G}x\|_2 \ge 2.$$

Now conditioning on this event, for an arbitrary $y \in \text{colspan}(A)$, $\|y\|_2 = 1$, we have

$$\begin{aligned}
\|\boldsymbol{G}y\|_2 &= \|\boldsymbol{G}(x_y + (y - x_y))\|_2 \\
&\ge \|\boldsymbol{G}x_y\|_2 - \|\boldsymbol{G}(y - x_y)\|_2 \\
&\ge 2 - \|\boldsymbol{G}\|_2\|y - x_y\|_2 \\
&\ge 1
\end{aligned}$$

as the net is chosen so that $\|y - x_y\|_2 \cdot \|G\|_2 \le 1$.

Conditioned on the event $\mathscr{E}$, we have that each row of G has at most $\mathrm{epll}(k)$ nonzero entries. Thus, each row of the matrix GA can be computed in $k \cdot \mathrm{epll}(k)$ time and hence the matrix GA can be computed in time $k^2\,\mathrm{epll}(k)$. As $\mathbf{Pr}[\mathscr{E}] \ge 99/100$, the claim follows. □

Theorem 3.4.3 (Subspace Embedding)**.** *Given an $n \times k$ matrix A, we can compute an $m \times k$ matrix SA with $m = k \cdot \mathrm{poly}(\log\log(k))$ such that with probability $\ge 9/10$, for all vectors $x \in \mathbb{R}^k$,*

$$\|Ax\|_2 \le \|SAx\|_2 \le \mathrm{epll}(k)\|Ax\|_2.$$

The matrix $S \cdot A$ can be computed in time $O(\mathrm{nnz}(A) + k^{2.1+o(1)})$ or more generally in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + k^{2+\gamma+o(1)})$ for any constant $\gamma > 0$. Further, for any matrix M with n rows,

$$\mathbf{E}[\|SM\|_F^2] \le \mathrm{epll}(k)\|M\|_F^2.$$

Proof. The matrix S is defined as follows

$$S = 2 \cdot G \cdot \mathscr{F} \cdot S_2 \cdot S_1$$

where S_1 is OSNAP for k dimensional subspaces with $\gamma = 0.1$, S_2 is OSNAP for k dimensional subspaces with $\gamma = 1/\log(k)$, $\mathscr{F}$ is Indyk's embedding for $O(k\log(k))$ dimensional subspaces as in Theorem 3.4.1 and G is the sparse embedding matrix with $k \cdot \mathrm{poly}(\log\log(k))$ rows as in Theorem 3.4.2. We have with probability $\ge 9/10$, for any vector $x \in \mathbb{R}^k$,

$$\frac{1}{2}\|Ax\|_2 \le \|S_2 \cdot S_1 \cdot Ax\|_2 \le \frac{3}{2}\|Ax\|_2.$$

Condition on the above event. From Theorem 3.4.1, we have

$$\frac{1}{2}\|Ax\|_2 \le \|S_2 \cdot S_1 \cdot Ax\|_2 = \|\mathscr{F} \cdot S_2 \cdot S_1 \cdot Ax\|_2 = \|S_2 \cdot S_1 \cdot Ax\|_2 \le \frac{3}{2}\|Ax\|_2.$$

By Theorem 3.4.1, every unit vector in the span of $\mathscr{F}$ has at least $k/\mathrm{polylog}(k)$ coordinates with an absolute value of at least $1/(\sqrt{k} \cdot \mathrm{epll}(k))$. Thus, the matrix $\mathscr{F} \cdot S_2 \cdot S_1 \cdot A$ satisfies the conditions of Theorem 3.4.2. Therefore with probability $\ge 9/10$, we have for all vectors $x \in \mathbb{R}^k$,

$$\|G \cdot \mathscr{F} \cdot S_2 \cdot S_1 \cdot Ax\|_2 \le \mathrm{epll}(k)\|\mathscr{F} \cdot S_2 \cdot S_1 \cdot Ax\|_2 \le \mathrm{epll}(k)\|Ax\|_2$$

and

$$\|G \cdot \mathscr{F} \cdot S_2 \cdot S_1 \cdot Ax\|_2 \ge \|\mathscr{F} \cdot S_2 \cdot S_1 \cdot Ax\|_2 \ge \frac{1}{2}\|Ax\|_2.$$

Thus with probability $\geq 8/10$, for all vectors x,

$$\|Ax\|_2 \leq \|S \cdot Ax\|_2 \leq \mathrm{epll}(k)\|Ax\|_2.$$

The matrix $S \cdot A$ can be computed as $2G(\mathcal{F}(S_2(S_1A)))$ in time

$$O(\mathrm{nnz}(A) + k^{2.1}\log^2(k) + k^{2+o(1)} + k^2 \cdot \mathrm{epll}(k))$$

where the last term follows from the fact that each of the $k\,\mathrm{poly}(\log\log(k))$ rows of the matrix G has at most $\mathrm{epll}(k)$ nonzero entries.

There is nothing special about $\gamma = 0.1$. We can choose any constant $1 > \gamma > 0$ and use OSNAP with the parameter γ which gives an overall running time of $O(\gamma^{-1}\,\mathrm{nnz}(A) + k^{2+\gamma+o(1)})$.

We now bound $\mathbf{E}_S[\|SM\|_F^2]$ for an arbitrary matrix M. We have

$$\begin{aligned}\mathbf{E}_S[\|SM\|_F^2] &= 4\,\mathbf{E}_{G,S_2,S_1}[\|G \cdot \mathcal{F} \cdot S_1 \cdot S_2M\|_F^2]\\ &\leq 4 \cdot \mathbf{E}_{S_1}[\mathbf{E}_{S_2}[\mathbf{E}_G[\|G \cdot \mathcal{F} \cdot S_2 \cdot S_1M\|_F^2 \mid S_1, S_2] \mid S_1]].\end{aligned}$$

First, $\mathbf{E}_G[\|G \cdot \mathcal{F} \cdot S_2 \cdot S_1M\|_F^2 \mid S_1, S_2] \leq Mp \cdot (\mathrm{scale}) \cdot \|\mathcal{F} \cdot S_2 \cdot S_1M\|_F^2$, where M is the number of rows of G, p is the probability of an entry of G being nonzero and scale $= \mathrm{epll}(k)$ is the scaling factor for the random sign matrix. As $M = k \cdot \mathrm{poly}(\log\log(k))$ and $p = \mathrm{epll}(k)/k$, we have $\mathbf{E}_G[\|G \cdot \mathcal{F} \cdot S_2 \cdot S_1M\|_F^2 \mid S_1, S_2] \leq \mathrm{epll}(k) \cdot \|\mathcal{F} \cdot S_2 \cdot S_1M\|_F^2 \leq \mathrm{epll}(k)\|S_2 \cdot S_1M\|_F^2$ as the matrix $\mathcal{F}$ does not change the euclidean norm of any vector. Thus,

$$\mathbf{E}_S[\|SM\|_F^2] \leq \mathrm{epll}(k)\,\mathbf{E}_{S_1}[\mathbf{E}_{S_2}[\|S_2 \cdot S_1M\|_F^2 \mid S_1]] \leq \mathrm{epll}(k)\|M\|_F^2,$$

where the last inequality follows from the fact that $\|S_iM\|_F^2$ is an unbiased estimator to $\|M\|_F^2$ if S_i is an OSNAP. □

3.5 Applications

3.5.1 Subspace Embeddings

We use the fast subspace embedding construction from the previous section to compute approximate leverage scores and then sample rows using the approximate leverage scores to compute $1+\varepsilon$ subspace embeddings in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + \varepsilon^{-3}n^{\gamma}k^{2+o(1)} + k^{\omega}\,\mathrm{poly}(\log\log(k)))$ for any constant γ. We then compose with an OSNAP to obtain a subspace embedding with $O(\varepsilon^{-2}k\log(k))$ rows.

Theorem 3.5.1 (Leverage Score Sampling)**.** *Given a full column rank matrix $A \in \mathbb{R}^{n\times k}$, let $\tau_i(A)$ for $i \in [n]$ be the leverage score of the i-th row. Let $p \in [0,1]^n$ be a vector of probabilities such that for all $i \in [n]$, $\min(1, r \cdot (\tau_i(A)/k)) \geq p_i \geq \min(1, r \cdot \beta \cdot (\tau_i(A)/k))$ for some $\beta < 1$, and let the $n \times n$ diagonal*

Algorithm 3.2: LEVERAGESCORESAMPLING

```
    Input: $A \in \mathbb{R}^{n\times k}$, $\varepsilon, \gamma > 0$
    Output: An $\varepsilon$ subspace embedding $\boldsymbol{S}_{\mathrm{lev}}A$
1   $SA \leftarrow$ SPARSEEMBEDDING$(A)$
2   $[Q, R^{-1}] \leftarrow$ QR-DECOMPOSITION$(SA)$                    // $QR^{-1} = SA$
3   $s \leftarrow k\exp(\mathrm{poly}(\log\log k)/\varepsilon^2$
4   $S_1 \subseteq [n], f_i$ for $i \in [S_1] \leftarrow$ SAMPLEFROMPRODUCT$(A, R, s, \gamma)$   // Lemma 3.5.3
5   For $i \in S_1$, set $(\boldsymbol{S}_{\mathrm{lev}})_{ii}$ to be equal to $1/\sqrt{f_i}$
6   return $\boldsymbol{S}_{\mathrm{lev}}A$ after removing 0-value rows
```

random matrix $\boldsymbol{S}_{\mathrm{lev}}$ be defined as follows: for each $i \in [n]$, the entry $(\boldsymbol{S}_{\mathrm{lev}})_{ii}$ is set to be equal to $1/\sqrt{p_i}$ with probability p_i, and is set to be 0 with probability $1 - p_i$. If $r \ge Ck\log(k)/\beta\varepsilon^2$ for an absolute constant C, then with probability $\ge 99/100$, for all vectors $x \in \mathbb{R}^d$

$$\|\boldsymbol{S}_{\mathrm{lev}}Ax\|_2^2 \in (1 \pm \varepsilon)\|Ax\|_2^2.$$

With probability $\ge 1 - \exp(-\Theta(k))$, the matrix $\boldsymbol{S}_{\mathrm{lev}}$ has at most $\Theta(Ck\log(k)/\beta\varepsilon^2)$ nonzero entries.

The following lemma shows that a subspace embedding S for the column space of a matrix A can be used to compute approximate leverage scores which can be used to perform leverage score sampling as described above to obtain a $1 + \varepsilon$ subspace embedding.

Lemma 3.5.2. *If S is a β subspace embedding for the column space of a full rank matrix $A \in \mathbb{R}^{n\times k}$ i.e., for any vector x,*

$$\|Ax\|_2 \le \|SAx\|_2 \le \beta\|Ax\|_2$$

and if $SA = QR^{-1}$ for an orthonormal matrix Q, then for all $i \in [n]$,

$$\tau_i(A)/\beta^2 \le \|A_{i*}R\|_2^2 \le \tau_i(A),$$

where $\tau_i(A)$ is the leverage score of the i-th row of A.

Proof. Let $AR = UT$ where U is an orthonormal matrix. As $\mathrm{colspan}(AR) = \mathrm{colspan}(A)$, we have that $\ell_i^2 = \|U_{i*}\|_2^2$. We first have for any vector x,

$$\|Tx\|_2 = \|UTx\|_2 = \|ARx\|_2 \le \|SARx\|_2 = \|Qx\|_2 = \|x\|_2$$

and

$$\|Tx\|_2 = \|UTx\|_2 = \|ARx\|_2 \ge (1/\beta)\|SARx\|_2 = (1/\beta)\|Qx\|_2 = (1/\beta)\|x\|_2.$$

Here we repeatedly used the facts that Q and U are orthonormal matrices. Thus, we obtain $\|T\|_2 \le 1$

and $\sigma_{\min}(T) \ge 1/\beta$. As $A_{i*}R = U_{i*}T$, we obtain that

$$\|A_{i*}R\|_2 = \|U_{i*}T\|_2 \le \|U_{i*}\|_2\|T\|_2 \le \|U_{i*}\|_2$$

and

$$\|A_{i*}R\|_2 = \|U_{i*}T\|_2 \ge \|U_{i*}\|_2\sigma_{\min}(T) \ge (1/\beta)\|U_{i*}\|_2.$$

Thus, ${\ell_i}^2/\beta^2 \le \|A_{i*}R\|_2^2 \le \ell_i^2$. □

Using our fast subspace embedding with $k\,\text{poly}(\log\log(k))$ rows and $\beta = \text{epll}(k)$, the above lemma shows that if we can compute the values $\|A_{i*}R\|_2^2$, then we can obtain a $1+\varepsilon$ subspace embedding with $k \cdot \text{epll}(k)/\varepsilon^2$ rows.

Often, the row norms $\|A_{i*}R\|_2^2$ are approximated with $\|A_{i*}R\boldsymbol{G}\|_2^2$, where $\boldsymbol{G}$ is a Gaussian matrix with $O(\log n)$ columns using the fact that for an arbitrary vector x, $\|x^{\mathrm{T}}\boldsymbol{G}\|_2^2 \in (1/2, 2)\|x\|_2^2$ with probability $1 - 1/\text{poly}(n)$. However, computing the matrix $AR\boldsymbol{G}$ takes $O((\text{nnz}(A) + k^2)\log(n))$ time.

The following simple lemma shows that instead of obtaining constant approximations to $\|A_{i*}R\|_2^2$ for all the rows by using a Gaussian matrix $\boldsymbol{G}$ with $O(\log(n))$ columns, we can use a Gaussian matrix $\boldsymbol{G}'$ with only $O(1/\gamma)$ columns to obtain $O(n^\gamma\log(n))$ factor approximations to $\|A_{i*}R\|_2^2$. We sample the rows using these coarse approximations and then compute constant-factor approximations to $\|A_{i*}R\|_2^2$ only for the rows that are sampled in the first stage and then reject each of the sampled rows with appropriate probabilities to obtain a leverage score sample.

Lemma 3.5.3. *Let $A \in \mathbb{R}^{n\times d}$ and $R \in \mathbb{R}^{d\times d}$ be such that for any vector $x \in \mathbb{R}^d$, the matrix-vector products ARx, Rx can be computed in time at most T_1 and T_2 respectively. Given parameters γ and s, there is an algorithm conditioned on an event $\mathscr{E}$, $\mathbf{Pr}[\mathscr{E}] \ge 95/100$, that samples indices $i \in [n]$ to obtain a random subset $\boldsymbol{S} \subseteq [n]$, such that each $i \in [n]$ is in the set $\boldsymbol{S}$ independently with probability f_i, where*

$$\min(1, s\frac{\|A_{i*}R\|_2^2}{\|AR\|_{\mathrm{F}}^2}) \ge f_i \ge \min(1, (s/16)\frac{\|A_{i*}R\|_2^2}{\|AR\|_{\mathrm{F}}^2}).$$

The algorithm returns the random subset $\boldsymbol{S}$ along with the probabilities f_i for $i \in \boldsymbol{S}$. The algorithm runs in time $O(\gamma^{-1}T_1 + T_2\log(n) + sdn^\gamma\log^2(n))$.

Proof. Let $p_i \coloneqq \|A_{i*}R\|_2^2/\|AR\|_{\mathrm{F}}^2$ for $i \in [n]$. Let $\boldsymbol{G}_1$ be a Gaussian matrix with $O(1)$ rows and n columns and $\boldsymbol{G}_2$ be a Gaussian matrix with d rows and $O(1)$ columns. We have

$$\frac{1}{2}\|AR\|_{\mathrm{F}}^2 \le \|\boldsymbol{G}_1AR\boldsymbol{G}_2\|_{\mathrm{F}}^2 \le 2\|AR\|_{\mathrm{F}}^2 \quad (\text{Event } \mathscr{E}_1)$$

with probability $\ge 99/100$. The matrix $\boldsymbol{G}_1AR\boldsymbol{G}_2$ can be computed in $O(T_1 + n)$ time. Let $\boldsymbol{G}_3$ be a

Gaussian matrix with $O(\log(n))$ columns. With probability $\ge 99/100$,

$$\text{for all } i \in [n], \quad \frac{1}{2}\|A_{i*}R\|_2^2 \le \|A_{i*}RG_3\|_2^2 \le 2\|A_{i*}R\|_2^2 \quad (\text{Event } \mathscr{E}_2).$$

We note that we *do not* compute the matrix ARG_3 but we only compute the matrix RG_3 which can be done in time $O(T_2\log(n))$.

Let G_4 be a Gaussian matrix with $t = O(1/\gamma)$ columns. Let $g_1, g_2, \ldots, g_t$ be the columns of the matrix G_4. For each $i \in [n]$, with probability $\ge 1 - 1/100n^2$, $\max_{j\in[t]} |\langle A_{i*}R, g_j\rangle| \ge \|A_{i*}R\|_2/n^{\gamma/2}$ using the fact that $\langle A_{i*}R, g_j\rangle_{j\in[t]}$ are independent Gaussians with standard deviation $\|A_{i*}R\|_2$. By a union bound, with probability $\ge 1 - 1/100n$, for all $i \in [n]$, we have

$$\|A_{i*}RG_4\|_2^2 \ge \max_{j\in[t]}\langle A_{i*}R, g_j\rangle^2 \ge \|A_{i*}R\|_2^2/n^\gamma.$$

By Lemma 1 of [LM00], we also obtain that with probability $\ge 1-1/100n$, for all $i \in [n]$, $\|A_{i*}RG_4\|_2^2 \le O(\log(n))\|A_{i*}R\|_2^2$. Thus, with probability $\ge 1 - 2/100n$, for all $i \in [n]$:

$$\frac{\|A_{i*}R\|_2^2}{n^\gamma} \le \|A_{i*}RG_4\|_2^2 \le C\log(n)\|A_{i*}R\|_2^2 \quad (\text{Event } \mathscr{E}_3).$$

We compute ARG_4 and all squared row norms $\|A_{i*}RG_4\|_2^2$ in time $O(T_1\gamma^{-1})$. Condition on the event $\mathscr{E} \coloneqq \mathscr{E}_1 \cap \mathscr{E}_2 \cap \mathscr{E}_3$. We have $\mathbf{Pr}[\mathscr{E}] \ge 95/100$.

Define $z_i \coloneqq 2n^\gamma\|A_{i*}RG_4\|_2^2/\|G_1ARG_2\|_F^2$. We have $4Cn^\gamma\log(n)p_i \ge z_i \ge p_i$ and define $q_i \coloneqq \min(1, sz_i)$. Sample $i \in [n]$ independently, each with probability q_i to obtain a random subset $S_1 \subseteq [n]$. If $i \in S_1$, compute the value $\|A_{i*}(RG_3)\|_2^2$ in time $O(d\log(n))$ and reject i with probability $1 - \min(1, (s/4)\|A_{i*}RG_3\|_2^2/\|G_1ARG_2\|_F^2)/q_i$.

We need to show that this procedure is well-defined. We have $(s/4)\|A_{i*}RG_3\|_2^2/\|G_1ARG_2\|_2^2 \le (s/4)(4p_i) = sp_i \le sz_i$ which implies that $\min(1, (s/4)\|A_{i*}RG_3\|_2^2/\|G_1ARG_2\|_F^2) \le q_i$ and therefore the rejection probability as defined is valid. Let S_2 be the subset obtained after performing the rejection step on S_1. The probability that a row $i \in S_2$ is

$$f_i = q_i \cdot \frac{\min(1, (s/4)\|A_{i*}RG_3\|_2^2/\|G_1ARG_2\|_F^2)}{q_i} \ge \min(1, (s/4)(p_i/4)) = \min(1, (s/16)p_i).$$

We also have that $f_i \le \min(1, sp_i)$. Thus, with probability $\exp(-s)$ only $O(s)$ rows survive the rejection.

Now, with probability $\ge 1 - \exp(-s)$, $|S_1| = O(\sum_i q_i) = O(sn^\gamma\log(n))$ and therefore the squared row norm $\|A_iRG_3\|_2^2$ has to be computed only for $O(sn^\gamma\log(n))$ rows. Thus, the time complexity of sampling is $O(\gamma^{-1}T_1 + T_2\log(n) + O(sdn^\gamma\log^2(n)))$. Therefore, conditioned on the event $\mathscr{E}$, the algorithm returns a subset $S \subseteq [n]$ sampled from the desired probability distribution in time $O(\gamma^{-1}T_1 + T_2\log(n) + sdn^\gamma\log^2(n))$. □

Using these lemmas, the following theorem shows that Algorithm 3.2 gives a $1+\varepsilon$ subspace embedding by sampling using approximate leverage scores.

Theorem 3.5.4. *Given a full rank matrix $A \in \mathbb{R}^{n\times k}$, a constant γ and a parameter $\varepsilon > 0$, we have the following:*

1. *Algorithm 3.2 computes a matrix $\boldsymbol{S}_{\text{lev}}A$ with $\Theta(\varepsilon^{-2}k\cdot \text{epll}(k))$ rows such that with probability $\geq 9/10$, for all vectors x,*
$$\|\boldsymbol{S}_{\text{lev}}Ax\|_2^2 \in (1\pm\varepsilon)\|Ax\|_2^2.$$
This matrix $\boldsymbol{S}_{\text{lev}}A$ can be computed in time
$$O(\gamma^{-1}\,\text{nnz}(A) + \varepsilon^{-2}n^{\gamma}k^{2+o(1)} + k^{\omega}\,\text{poly}(\log\log(k))).$$

2. *Composing $\boldsymbol{S}_{\text{lev}}$ with the matrix $\boldsymbol{S}_{\textsf{OSNAP}}$, an* **OSNAP** *with $O(\varepsilon^{-2}k\log(k))$ and at most $O(\varepsilon^{-1}\log(k))$ nonzero entries in each column, we obtain that with probability $\geq 9/10$, for all vectors x,*
$$\|\boldsymbol{S}_{\textsf{OSNAP}}\cdot \boldsymbol{S}_{\text{lev}}\cdot Ax\|_2^2 \in (1\pm O(\varepsilon))\|Ax\|_2^2.$$
The matrix $\boldsymbol{S}_{\textsf{OSNAP}}\cdot(\boldsymbol{S}_{\text{lev}}A)$ can be computed in time $O(\varepsilon^{-3}k^{2+o(1)})$ and hence, overall, the matrix $\boldsymbol{S}_{\textsf{OSNAP}}\cdot \boldsymbol{S}_{\text{lev}}\cdot A$ can be computed in time
$$O(\gamma^{-1}\,\text{nnz}(A) + k^{\omega}\,\text{poly}(\log\log(k)) + \varepsilon^{-3}k^{2+o(1)} + \varepsilon^{-2}n^{\gamma+o(1)}k^{2+o(1)})$$
for any constant γ.

Proof. From Theorem 3.4.2, we have a subspace embedding $\boldsymbol{S}_{\text{fast}}$ with $O(k\,\text{poly}(\log\log k))$ rows and distortion $\text{epll}(k)$ that can be applied to matrix A in time $O(\gamma^{-1}\,\text{nnz}(A)+k^{2+\gamma+o(1)})$ for any constant $\gamma > 0$. Compute the matrices Q, R^{-1} such that Q has orthonormal columns and $\boldsymbol{S}_{\text{fast}}A = QR^{-1}$ which can be done in time $O(k^{\omega}\,\text{poly}(\log\log(k)))$. By Lemma 3.5.2, we have
$$\frac{\tau_i(A)}{\text{epll}(k)} \leq \|A_{i*}R\|_2^2 \leq \tau_i(A)$$
which implies, using the fact $\sum_i \tau_i(A) = k$, that
$$\frac{\tau_i(A)}{k\cdot\text{epll}(k)} \leq \frac{\|A_{i*}R\|_2^2}{\|AR\|_{\text{F}}^2}.$$
Using Lemma 3.5.3, conditioned on the event $\mathscr{E}$, we can sample a random subset $\boldsymbol{S}$ along with probabilities f_i for $i\in\boldsymbol{S}$ such that each $i\in[n]$ is independently in the subset $\boldsymbol{S}$ with probability f_i,
$$f_i \geq \min(1, (s/4)\cdot\frac{\|A_{i*}R\|_2^2}{\|AR\|_{\text{F}}^2}) \geq \min(1, (s/4)\cdot\frac{\tau_i(A)}{k\cdot\text{epll}(k)}).$$

For $s = \Theta(k\log(k)\operatorname{epll}(k)/\varepsilon^2)$, we have $f_i \geq \min(1, C\tau_i(A)\log(k)/\varepsilon^2)$ which implies that the matrix $\mathbf{S}_{\text{lev}}$ constructed by Algorithm 3.2 is a $1+\varepsilon$ subspace embedding, with probability $\geq 9/10$, for the column space of A by Theorem 3.5.1. In the notation of Lemma 3.5.3, for the matrices A and R, $T_1 = \operatorname{nnz}(A) + k^2$ and $T_2 = k^2$. Thus, the sampling process runs in time

$$O(\gamma^{-1}\operatorname{nnz}(A) + k^2\log(n) + \varepsilon^{-2}n^{\gamma}k^2\exp(\operatorname{poly}(\log\log k))) = O(\gamma^{-1}\operatorname{nnz}(A) + \varepsilon^{-2}n^{\gamma+o(1)}k^{2+o(1)}).$$

Thus, overall, in time $O(\gamma^{-1}\operatorname{nnz}(A) + \varepsilon^{-2}n^{\gamma+o(1)}k^{2+o(1)} + k^{\omega}\operatorname{poly}(\log\log k))$, we can compute a leverage score sampling matrix $\mathbf{S}_{\text{lev}}$ with $O(\varepsilon^{-2}k\exp(\log\log k))$ rows such that for all $x \in \mathbb{R}^k$,

$$\|\mathbf{S}_{\text{lev}}Ax\|_2^2 \in (1\pm\varepsilon)\|Ax\|_2^2.$$

As $\operatorname{nnz}(\mathbf{S}_{\text{lev}}A) \leq (\varepsilon^{-1}k)^2\operatorname{epll}(k)$, the OSNAP embedding $\mathbf{S}_{\mathsf{OSNAP}}$ can be applied to $\mathbf{S}_{\text{lev}}A$ in $O(\varepsilon^{-3}k^2\operatorname{epll}(k))$ time and the fact that $\mathbf{S}_{\mathsf{OSNAP}}\cdot\mathbf{S}_{\text{lev}}$ is a subspace embedding follows from the composability. Thus, we can compute $\mathbf{S}_{\mathsf{OSNAP}}\cdot\mathbf{S}_{\text{lev}}\cdot A$ which has $O(\varepsilon^{-2}k\log k)$ rows in $O(\gamma^{-1}\operatorname{nnz}(A) + k^{\omega}\operatorname{poly}(\log\log k) + \varepsilon^{-3}k^{2+o(1)} + \varepsilon^{-2}n^{\gamma+o(1)}k^{2+o(1)})$ time. □

3.5.2 Linear Regression

Let $A \in \mathbb{R}^{n\times k}$ and $b \in \mathbb{R}^n$. By the linear regression problem (A, b), we mean $\min_x \|Ax - b\|_2$ and $\text{OPT}(A, b)$ denotes the optimum value of this problem. We prove the following theorem.

Theorem 3.5.5. *Given a full-rank matrix $A \in \mathbb{R}^{n\times k}$ and $b \in \mathbb{R}^n$, we obtain a solution x^* such that*

$$\|Ax^* - b\|_2 \leq (1+\varepsilon)\text{OPT}(A, b)$$

in time $O(\gamma^{-1}\operatorname{nnz}(A) + \varepsilon^{-3}n^{\gamma+o(1)}k^{2+o(1)} + k^{\omega}\operatorname{poly}(\log\log k))$ for any constant γ.

Proof. We first find a $1+\varepsilon$ subspace embedding $\mathbf{S}$ for the column space of $[A, b]$. From Theorem 3.5.4, $\mathbf{S}A$ and $\mathbf{S}b$ can be computed in at most $O(\gamma^{-1}\operatorname{nnz}(A) + \varepsilon^{-3}n^{\gamma+o(1)}k^{2+o(1)} + k^{\omega}\operatorname{poly}(\log\log k))$ time. We can also compute a preconditioner R using the fast subspace embedding from Theorem 3.4.2 such that

$$\kappa(AR) = \operatorname{epll}(k)$$

by first computing $\mathbf{S}_{\text{fast}}A = QR^{-1}$ and then inverting R^{-1} to obtain R. The matrix R can be computed in time $O(\gamma^{-1}\operatorname{nnz}(A) + k^{2+\gamma+o(1)} + k^{\omega}\operatorname{poly}(\log\log(k)))$ for any constant γ. We also have that

$$\kappa(\mathbf{S}AR) = \operatorname{epll}(k).$$

Let x^* be a solution such that $\|SARx^* - Sb\|_2 \le (1+\varepsilon)\min_x \|SARx - Sb\|_2$. Then, we have

$$\|ARx^* - b\|_2 \le \frac{1}{1-\varepsilon}\|SARx^* - Sb\|_2 \le \frac{1+\varepsilon}{1-\varepsilon}\|SAx_{\mathrm{opt}} - Sb\|_2 \le \frac{(1+\varepsilon)^2}{1-\varepsilon}\|Ax_{\mathrm{opt}} - b\|_2.$$

Thus, Rx^* is a $1+O(\varepsilon)$ approximate solution for the linear regression problem (A, b). Now, we focus on obtaining a $1+\varepsilon$ approximate solution for the regression problem (SAR, Sb).

We first compute an approximate solution for the regression problem as follows: let S_{fast} be the subspace embedding with $k\,\mathrm{poly}(\log\log(k))$ rows for the column space of $[A, b]$. Let $x^{(0)} = (S_{\mathrm{fast}}A)^+(S_{\mathrm{fast}}b)$. This solution can be computed in time $O(\mathrm{nnz}(A)+k^{2+\gamma+o(1)}+k^{\omega}\,\mathrm{poly}(\log\log(k)))$. Let $x_{\mathrm{start}} = R^{-1}x^{(0)}$ which can also be computed in time $O(k^2)$. Now, we have

$$\|SARx_{\mathrm{start}} - Sb\|_2 \le (1+\varepsilon)\|ARx_{\mathrm{start}} - b\|_2 = (1+\varepsilon)\|Ax^{(0)} - b\|_2 \le (1+\varepsilon)\|S_{\mathrm{fast}}Ax^{(0)} - S_{\mathrm{fast}}b\|_2.$$

Let x_S be the optimal solution for the regression problem (SA, Sb). By optimality of $x^{(0)}$ for the regression problem $(S_{\mathrm{fast}}A, S_{\mathrm{fast}}b)$, we have

$$\begin{aligned}\|SARx_{\mathrm{start}} - Sb\|_2 &\le (1+\varepsilon)\|S_{\mathrm{fast}}Ax^{(0)} - S_{\mathrm{fast}}b\|_2\\ &\le (1+\varepsilon)\|S_{\mathrm{fast}}Ax_S - S_{\mathrm{fast}}b\|_2\\ &\le (1+\varepsilon)\cdot \mathrm{epll}(k)\cdot \|Ax_S - b\|_2\\ &\le \mathrm{epll}(k)\cdot \mathrm{OPT}((SA, Sb)).\end{aligned}$$

Thus, x_{start} is an $\mathrm{epll}(k)$ approximate solution for the linear regression problem (SAR, Sb). Using the solution x_{start}, we can obtain a $1+\varepsilon$ approximate solution in $O(\mathrm{epll}(k)/\varepsilon)$ iterations of gradient descent where each iteration can be performed in time $O(k^2\log(k)/\varepsilon^2)$. Thus, overall, in time

$$O(\gamma^{-1}\,\mathrm{nnz}(A) + \varepsilon^{-3}n^{\gamma+o(1)}k^{2+o(1)} + k^{\omega}\,\mathrm{poly}(\log\log k)),$$

we can compute a $1+O(\varepsilon)$ approximate solution for the linear regression problem (A, b). □

3.5.3 Rank Computation and Independent Row Selection

We give an algorithm to compute a maximal set of independent rows of an $n \times n$ matrix A of rank $k = n^{\Omega(1)}$ in time $O(\gamma^{-1}\,\mathrm{nnz}(A)+k^{2+\gamma+o(1)}+k^{\omega}\,\mathrm{poly}(\log\log(k)))$ for any constant $\gamma > 0$, improving upon the earlier running time of $O((\mathrm{nnz}(A)+k^{\omega})\log(k))$ from [CKL13] for any constant $\omega > 2$.

Definition 3.5.6 (Rank Preserving Sketches). A distribution $\mathcal{S}$ over $z_S \times n$ matrices is a rank preserving sketch if there exists a constant c such that for $S \sim \mathcal{S}$, with high probability, for a given matrix $A \in \mathbb{R}^{n\times d}$, $\min(\mathrm{rank}(SA), z_S/c) = \min(\mathrm{rank}(A), z_S/c)$ i.e., multiplying A with the matrix S preserves the rank if $\mathrm{rank}(A) \le z_S/c$.

Theorem 3.5.7 ([CKL13]). *There are rank-preserving sketching distributions as above with $c = 11$ such that*

- *SA can be computed in $O(\text{nnz}(A))$ time*
- *S has at most 2 nonzero entries in a column*
- *S has at most $2n/z_S$ nonzero entries in a row*

They use rank preserving sketches to give an algorithm to compute the rank of an arbitrary matrix and an algorithm to compute a maximal set of linearly independent rows of the matrix.

Theorem 3.5.8 (Theorem 2.6 of [CKL13]). *Let $A \in \mathbb{R}^{n\times d}$ be an arbitrary matrix with $n \geq d$. There is a randomized algorithm to compute $k = \text{rank}(A)$ in time $O(\text{nnz}(A)\log(k) + \min(k^\omega, k \cdot \text{nnz}(A)))$ with failure probability at most $O(1/n^{1/3})$. There is also an algorithm to find k linearly independent rows of the matrix A in time $O((\text{nnz}(A) + k^\omega)\log(n))$ with failure probability at most $O(\log(n)/n^{1/3})$.*

We show that the $\log(k)$ factor can be removed from the time required to compute the rank of the matrix.

Theorem 3.5.9 (Rank computation). *Given $A \in \mathbb{R}^{n\times d}$, let $k = \text{rank}(A)$. Let ω be the matrix multiplication constant and assume $\omega > 2$. Consider two cases:*

1. *If $k \leq \log(n)^{2/(\omega-2)}$, k can be computed in time $O(\text{nnz}(A) + \log(n)^{6/(\omega-2)}) = O(\text{nnz}(A))$.*
2. *If $k \geq \log(n)^{2/(\omega-2)}$, k can be computed using Algorithm 3.3 (Rank) in time $O(\text{nnz}(A)+\min(k^\omega, k\cdot \text{nnz}(A)))$.*

Proof. If $k \leq \log(n)^{2/(\omega-2)}$, then we have rank preserving sketches S, R such that SAR can be computed in time $\text{nnz}(A)$, SAR is an $O(\log(n)^{2/(\omega-2)}) \times O(\log(n)^{2/(\omega-2)})$ matrix and $\text{rank}(SAR) = \text{rank}(A)$. Now the rank of SAR can be computed in time $O(\log(n)^{6/(\omega-2)})$. Thus, $\text{rank}(A)$ can be computed in time $O(\text{nnz}(A) + \log(n)^{6/(\omega-2)})$.

In the case of $k \geq \log(n)^{2/(\omega-2)}$, consider Algorithm 3.3. As $z \geq \Theta(\sqrt{n/\log(n)})$, with failure probability at most $\Theta(\sqrt{\log(n)/n})$, the sketch SAR is rank preserving. As SAR is a $z \times z$ matrix, we have $\text{nnz}(SAR) \leq z^2 \leq O(\text{nnz}(A)/\log(n))$. So, the rank k_1 of SAR can be computed in time $O(\text{nnz}(SAR)\log(k_1) + \min(k_1^\omega, k_1 \cdot \text{nnz}(SAR))$ by Theorem 3.5.8. As $k_1 \leq k$, we have that the rank k_1 can be computed in time $O(\text{nnz}(A) + \min(k^\omega, k \cdot \text{nnz}(A)))$.

We now have two cases. In the case that $k_1 < (\text{nnz}(A)/\log(n))^{1/2}$, as we have

$$\min(\text{rank}(A), (\text{nnz}(A)/\log(n))^{1/2}) = \min(\text{rank}(S_1AR_1), (\text{nnz}(A)/\log(n))^{1/2}),$$

we obtain that $\text{rank}(A) = \text{rank}(SAR) = k_1$.

If $(\text{nnz}(A)/\log(n))^{1/2} \leq k_1$, we have $k = \text{rank}(A) \geq k_1 \geq (\text{nnz}(A)/\log n)^{1/2}$ which shows that $\text{nnz}(A)\log(n) \leq k^2\log^2(n) \leq k^\omega$ for any $\omega > 2$ and $k \geq \log(n)^{2/(\omega-2)}$. We can now compute $\text{rank}(A)$ in time $O(\text{nnz}(A)\log(k) + \min(k^\omega, k\cdot\text{nnz}(A)))$ by Theorem 3.5.8. As $\text{nnz}(A)\log(k) = O(\min(\text{nnz}(A)\cdot k, k^\omega))$, we obtain that the running time is $O(\text{nnz}(A) + \min(k^\omega, k \cdot \text{nnz}(A)))$. □

Algorithm 3.3: RANK(A)

```
   Input: $A \in \mathbb{R}^{n \times d}$, $\mathrm{rank}(A) \geq (\log(n))^{6/(\omega-2)}$
   Output: $k \coloneqq \mathrm{rank}(A)$
   // CKL-RE, the algorithm of Theorem 2.6 of [CKL13]
 1 $z \leftarrow c \cdot (\mathrm{nnz}(A)/\log n)^{1/2}$                // $c \geq 1$ is a constant
 2 Generate rank-preserving sketches $S \in \mathbb{R}^{z \times n}$ and $R^{\mathrm{T}} \in \mathbb{R}^{z \times d}$
 3 Compute $SAR$                                  // using Theorem 3.5.7
 4 $k_1 \leftarrow \mathrm{rank}(SAR)$                           // using CKL-RE
 5 if $k_1 < z/c$ then
 6     return $k_1$
 7 end
 8 $k_2 \leftarrow \mathrm{rank}(A)$                             // using CKL-RE
 9 return $k_2$
```

We now describe an algorithm to compute k linearly independent rows of a matrix $A \in \mathbb{R}^{n \times d}$ of rank k in time $O(\mathrm{nnz}(A) + k^{\omega} \operatorname{poly}(\log\log(n)))$, replacing the $\log(n)$ factor in the running time of [CKL13] with $\operatorname{poly}(\log\log(n))$. Thus for matrices A with $k^{\omega-1} \leq \mathrm{nnz}(A) \leq k^{\omega}/\log(n)$, we can now compute the rank k and a set of k linearly independent rows in time $O(k^{\omega} \operatorname{poly}(\log\log(k)))$ instead of $O(k^{\omega}\log(k))$ time.

Without loss of generality, using the rank-preserving sketch, we can assume that $d = ck$ for a constant c. The following lemma describes a reduction to a sparse sub-matrix of A which also has rank equal to $\mathrm{rank}(A)$.

Algorithm 3.4: ROWREDUCTION(A, k)

```
   Input: $A \in \mathbb{R}^{n \times ck}$, $\mathrm{rank}(A) = k$
   Output: $A_Q \in \mathbb{R}^{m \times ck}$, $m \leq (3n/11)k$, $\mathrm{nnz}(A_Q) \leq \max((2/5)\,\mathrm{nnz}(A), \Theta(k^2))$, $\mathrm{rank}(A_Q) =$
           $k$
 1 $\boldsymbol{S} \leftarrow \mathbb{R}^{ck \times n}$ be a rank-preserving sketch
 2 Compute $\boldsymbol{S}A$
 3 Compute $P \subseteq [ck]$, $|P| = k$ such that $(\boldsymbol{S}A)_P$ has $k$ linearly independent rows
 4 Let $Q \leftarrow \{\, i \in [m] \mid \boldsymbol{S}_{ji} \neq 0 \text{ for some } j \in P \,\}$
 5 return $A_Q$
```

Lemma 3.5.10. *Let $A \in \mathbb{R}^{n \times ck}$ be an arbitrary matrix of rank k. There is a submatrix $A_Q \in \mathbb{R}^{m \times ck}$ that can be computed in time $O(\mathrm{nnz}(A) + k^{\omega})$ such that*

- $m = |Q| \leq \max(3n/11, O(k))$,
- $\mathrm{nnz}(A_Q) \leq \max((2/5) \cdot \mathrm{nnz}(A), \Theta(k^2))$, *and*
- $\mathrm{rank}(A_Q) = k$.

Algorithm 3.5: INDEPENDENTROWS(A, k)

```
    Input: $A \in \mathbb{R}^{n\times d}$, rank$(A) = k$
    Output: $A_Q \in \mathbb{R}^{k\times d}$, rank$(A_Q) = k$
1 $S \leftarrow \mathbb{R}^{ck\times d}$ be a rank preserving sketch
2 $B \leftarrow AS^{\mathrm{T}}$
3 Compute $B'$ by applying ROWREDUCTION $\Theta(\log\log(n))$ times
4 Compute $S_{\mathrm{lev}}$, a leverage score subspace embedding for $B'$ using Theorem 3.5.4 with
    $\gamma = 1/\log(n)$ and $\varepsilon = 0.1$
5 Compute $B''$ with $O(k)$ rows by applying ROWREDUCTION to the matrix $S_{\mathrm{lev}}A$,
    $\Theta(\log\log(k))$ times
6 Compute $k$ linearly independent rows of $B''$ and return $A_Q$ corresponding to these $k$ rows
```

Proof. Let $S \in \mathbb{R}^{ck\times n}$ be a rank-preserving sketch for $c = 11$. We have $\mathrm{rank}(SA) = \mathrm{rank}(A) = k$ with probability $\geq 1 - O(1/k)$. Consider a set L of k linearly independent rows of the matrix SA which can be determined in $O(k^{\omega})$ time[1]. Let $Q \subseteq [n]$ be the set of rows of A that contribute to the construction of the submatrix $(SA)_L$ which implies that $k \geq \mathrm{rank}(A_Q) \geq \mathrm{rank}((SA)_L) = k$ and hence $\mathrm{rank}(A_Q) = k$. We therefore have that the sub-matrix A_Q consists of k linearly independent rows. The reduction $A \to A_Q$ can be performed in $O(\mathrm{nnz}(A)+k^{\omega})$ time. As each row of the matrix S has at most $2n/11k$ nonzero entries, we have $|Q| \leq (2n/11k)\cdot k \leq 2n/11$. We now bound $\mathrm{nnz}(A_Q)$.

Let $P \subseteq [ck]$ be an arbitrary subset of size k. We show that if $Q_P \subseteq [n]$ is the subset of rows of A that contribute to the construction of the sub-matrix $(SA)_P$, then $\mathrm{nnz}(A_{Q_P}) \leq (2/5)\cdot \mathrm{nnz}(A)$ with high probability.

Let X_i be the random variable that indicates if A_{i*} contributes to the construction of $(SA)_P$ i.e., if $i \in Q_P$. By inspecting the proof of Theorem 3.5.7, we obtain that $\mathbf{Pr}[X_i = 0] = (1-1/c)^2$. Thus, for $c = 11$, we obtain that $\mathbf{Pr}[X_i = 1] = 1 - (1-1/11)^2 = 21/121$. We also note that the random variables $X_1, \ldots, X_n$ are negatively associated [Waj17]. Let a_i denote the number of nonzero entries of the row A_{i*} which implies that $\sum_i a_i = \mathrm{nnz}(A)$. Now, we have $\mathrm{nnz}(A_{Q_P}) = \sum_i a_i X_i$. Using the Chernoff-Hoeffding bound for negatively associated random variables [DR96],

$$\mathbf{Pr}[\mathrm{nnz}(A_{Q_P}) = \sum_i a_i X_i \geq \mathrm{nnz}(A)\cdot 21/121 + t] \leq 2\exp\left(-\frac{2t^2}{\sum_i a_i^2}\right).$$

By a union bound over all $\binom{11k}{k} \leq (11e)^k$ subsets P, we obtain that for a constant C,

$$\mathbf{Pr}[\text{There is a subset } P \subseteq [11k], |P| = k \text{ with } \mathrm{nnz}(A_{Q_P}) \geq \mathrm{nnz}(A)/5 + t] \leq 2\exp\left(Ck - \frac{2t^2}{\sum_i a_i^2}\right).$$

[1]Using a recursive algorithm that first finds the set of linearly independent rows in the top half of the matrix and projecting away the bottom half rows away from the top half.

Now, we have $\sum_i a_i^2 \leq \max_i a_i \cdot \sum_i a_i \leq 11k \cdot (\text{nnz}(A))$ since the matrix A is assumed to have only $ck = 11k$ columns. For $t \geq \Theta(k\sqrt{\text{nnz}(A)})$, we obtain that with probability $\geq 1 - \exp(-\Theta(k))$, for all $P \subseteq [11k], |P| = k$, we have that $\text{nnz}(A_{Q_P}) \leq \text{nnz}(A)/5 + t$. For $\text{nnz}(A) \geq \Theta(k^2)$, we have $\text{nnz}(A)/5 \geq \Theta(k\sqrt{\text{nnz}(A)})$ which implies that for all P, $\text{nnz}(A_{Q_P}) \leq (2/5)\,\text{nnz}(A)$. This, in particular, implies that for $M = Q_L$, that corresponds to the set of rows contributing to a linearly independent set of rows of (SA), we have $\text{nnz}(A_M) \leq (2/5) \cdot \text{nnz}(A)$ if $\text{nnz}(A) \geq \Theta(k^2)$. □

Recursively applying the above lemma, we obtain the following.

Corollary 3.5.11. *Let $A \in \mathbb{R}^{n\times d}$ be an arbitrary matrix of rank k. There is a matrix $A' \in \mathbb{R}^{m\times ck}$ with either $\text{nnz}(A') \leq \text{nnz}(A)/\log(n)$ or $\text{nnz}(A') \leq \Theta(k^2)$ such that*

- $\text{rank}(A') = \text{rank}(A) = k$, *and*
- $m \leq \max(n/\text{poly}(\log(n)), k)$
- *linearly independent rows of A' correspond to linearly independent rows of A.*

The reduction $A \longrightarrow A'$ can be performed in $O(\text{nnz}(A) + k^\omega \log\log(n))$ time.

Proof. Let $N = \Theta(\log\log(n))$ and $A^{(0)} = A$. Starting with $i = 0$, we apply the above reduction $A^{(i)} \longrightarrow A^{(i+1)}$ to obtain a matrix with $\text{nnz}(A^{(i+1)}) \leq (2/5) \cdot \text{nnz}(A^{(i)})$. Then

$$\text{nnz}(A^{(N)}) \leq \max((2/5)^N \,\text{nnz}(A), \Theta(k^2)) \leq \max(\text{nnz}(A)/\log(n), \Theta(k^2)).$$

The time complexity is $O(\sum_{i=1}^N(\text{nnz}(A^{(i)}) + k^\omega)) = O(\text{nnz}(A) + k^\omega \log\log(n))$. □

We have now reduced the general problem of computing k linearly independent rows of a rank-k $n \times d$ matrix A to computing k linearly independent rows of a rank-k $m \times ck$ matrix A' with $m \leq O(\max(k, n/\text{poly}(\log(n)))$ and $\text{nnz}(A') \leq O(\max(k^2, \text{nnz}(A)/\log(n)))$. Using these reductions, we have the following theorem.

Theorem 3.5.12. *Given an arbitrary matrix $A \in \mathbb{R}^{n\times d}$ of rank k, Algorithm 3.5 computes a set of k linearly independent rows of the matrix A in time $O(\text{nnz}(A) + k^\omega\,\text{poly}(\log\log(n)) + k^{2+o(1)})$.*

Proof. Let $S \in \mathbb{R}^{ck\times d}$ be a rank preserving sketch which implies $\text{rank}(AS^{\text{T}}) = \text{rank}(A) = k$ with probability $1 - O(1/k)$. Condition on this event. Let $M \subseteq [n]$, $|M| = k$ be such that rows of the sub-matrix $(AS^{\text{T}})_M = A_M S^{\text{T}}$ are linearly independent. Then, $k \geq \text{rank}(A_M) \geq \text{rank}(A_M S^{\text{T}}) = k$ which implies $\text{rank}(A_M) = k$. Thus, we only have to find k linearly independent rows of the $n \times ck$ matrix $B = AS^{\text{T}}$. We also have $\text{nnz}(B) = O(\text{nnz}(A))$. Using the above corollary, we can find an $m \times ck$ sub-matrix B' such that $\text{rank}(B') = k$, $\text{nnz}(B') \leq O(\max(\text{nnz}(B)/\text{poly}(\log(n)), \Theta(k^2))$ and $m = n/\text{poly}(\log(n))$.

From Theorem 3.5.4, using $\gamma = 1/\log(n)$, in time $O(\text{nnz}(B')\log(n) + k^\omega\,\text{poly}(\log\log(n)) + k^{2+o(1)} + m\gamma^{-1}) = O(\text{nnz}(A) + k^\omega\,\text{poly}(\log\log(n)) + k^{2+o(1)})$, we can compute a row sampling matrix S_{lev} that samples $O(k \cdot \text{epll}(k))$ rows such that

$$\|S_{\text{lev}} B' x\|_2^2 \in (1 \pm 1/10)\|B'x\|_2^2$$

for all vectors x. This, implies that the matrix $S_{\text{lev}} B'$ has rank k and hence has k linearly independent rows.

As S_{lev} is a leverage score sampling matrix, the rows of $S_{\text{lev}} B'$ are multiples of rows of the matrix B'. Thus, a set of k linearly independent rows of the matrix $S_{\text{lev}} B'$ directly corresponds to a set of k linearly independent rows of B which corresponds to a set of k linearly independent rows of the matrix A.

Applying the row reduction $\text{poly}(\log\log(k))$ times to the matrix $S_{\text{lev}} B'$, we obtain a matrix B'' of dimension $O(k) \times k$ from which we can determine a set of k linearly independent rows in time $O(k^{\omega})$. This concludes the proof. $\square$

3.5.4 Low-Rank Approximation

Let $A \in \mathbb{R}^{n \times d}$ be an arbitrary matrix. We want to compute a matrix B of rank at most k such that

$$\|A - B\|_{\text{F}}^2 \le (1 + \varepsilon)\|A - [A]_k\|_{\text{F}}^2.$$

Let OPT_A denote $\|A - [A]_k\|_{\text{F}}^2$. Our main theorem for Low-Rank Approximation (LRA) is as follows.

Theorem 3.5.13. *Let $A \in \mathbb{R}^{n \times d}$, $k < \min(n, d)$ be a rank parameter and $\varepsilon > 0$ be an accuracy parameter. There is an algorithm that outputs matrices $V \in \mathbb{R}^{n \times k}$ and $X \in \mathbb{R}^{k \times d}$, $V^{\text{T}} V = I_k$, such that with $\Omega(1)$ probability,*

$$\|A - VX\|_{\text{F}}^2 \le (1 + \varepsilon)\|A - [A]_k\|_{\text{F}}^2.$$

The algorithm runs in time $O(\gamma^{-1}\,\text{nnz}(A) + \varepsilon^{-1}(n + d)k^{\omega - 1} + \varepsilon^{-1}k(nd^{\gamma + o(1)} + dn^{\gamma + o(1)}) + \text{poly}(\varepsilon^{-1}k))$ for any constant $\gamma > 0$.

In the following sections, we will describe how to compute the left factor V and the right factor X. We are not very careful with probabilities, as we only have to condition over the success of $O(1)$ events, and all these events can be chosen to have a success probability $1 - c$ for any absolute constant $c > 0$ without affecting the time complexity.

We start with a residual sampling algorithm that lets us obtain a subspace containing a $1 + \varepsilon$ approximation given a subspace that is only $O(1)$ approximate.

Residual Sampling

Suppose we have a subspace $V \in \mathbb{R}^d$ such that

$$\|A - A\mathbb{P}_V\|_{\text{F}}^2 \le K\|A - [A]_k\|_{\text{F}}^2.$$

The following theorem of [DRVW06] shows that sampling $O(K \cdot k/\varepsilon)$ rows of the matrix A with probabilities proportional to the squared distances of the rows to the subspace V gives a subspace that along with V contains a $1+\varepsilon$ rank-k approximation to the matrix A.

Theorem 3.5.14 (Theorem 2.1 of [DRVW06])**.** *Let $A \in \mathbb{R}^{n\times d}$ and $V \subseteq \mathbb{R}^d$ be a subspace. Let $E = A - A\mathbb{P}_V$, the matrix formed by projecting each row of A away from the subspace V. Let $\boldsymbol{S}$ be a random sample of s rows of A from a distribution $\mathcal{D}$ such that row i is chosen with probability $p_i \ge \alpha\|E_{i*}\|_2^2/\|E\|_{\mathrm{F}}^2$. Then for any non-negative integer k,*

$$\mathbf{E}_{\boldsymbol{S}}[\min_{\substack{\text{rank-}k\ B\\ \mathit{rowspan}(B)\subseteq V+\mathit{rowspan}(A_{\boldsymbol{S}})}} \|A-B\|_{\mathrm{F}}^2] \le \|A-A_k\|_{\mathrm{F}}^2 + \frac{k}{s\alpha}\|E\|_{\mathrm{F}}^2.$$

Instead of sampling s rows independently from the distribution p, we can also sample each $i \in [n]$ with probability $q_i \coloneqq \min(1, sp_i)$ and obtain the same result for the resulting random subset of rows. Sampling each $i \in [n]$ independently with probability q_i lets us use the sampling framework from Lemma 3.5.3.

Lemma 3.5.15 (Sampling each row independently)**.** *Let $A \in \mathbb{R}^{n\times d}$ and V be a subspace in $\mathbb{R}^d$ and let $E = A - A\mathbb{P}_V$. Sample each $i \in [n]$ independently with a probability $q_i \coloneqq \min(1, sp_i)$, with $p_i \ge \alpha\|E_{i*}\|_2^2/\|E\|_{\mathrm{F}}^2$ to obtain a random subset $\boldsymbol{S} \subseteq [n]$. For any nonnegative integer k,*

$$\mathbf{E}_{\boldsymbol{S}}[\min_{\substack{\text{rank-}k\ B\\ \mathit{rowspan}(B)\subseteq V+\mathit{rowspan}(A_{\boldsymbol{S}})}} \|A-B\|_{\mathrm{F}}^2] \le \|A-A_k\|_{\mathrm{F}}^2 + \frac{k}{s\alpha}\|E\|_{\mathrm{F}}^2.$$

Proof. Let $u^{(1)}, \dots, u^{(d)}$ be the left singular vectors and $v^{(1)}, \dots, v^{(d)}$ be the right singular vectors of the matrix A. For $j = 1, \dots, k$, let

$$\boldsymbol{X}^{(j)} = \sum_{i:q_i<1} \frac{u_i^{(j)}}{q_i}(E_{i*})^{\mathrm{T}}\boldsymbol{I}[i \text{ is sampled}]$$

and $\boldsymbol{w}^{(j)} = \boldsymbol{X}^{(j)} + \sum_{i:q_i=1} u_i^{(j)}(E_{i*})^{\mathrm{T}} + \mathbb{P}_V A^{\mathrm{T}} u^{(j)}$. We have $\mathbf{E}[\boldsymbol{w}^{(j)}] = A^{\mathrm{T}}u^{(j)} = \sigma_j v^{(j)}$ and

$$\mathbf{E}[\|\boldsymbol{w}^{(j)} - \sigma_j v^{(j)}\|_2^2] = \mathbf{E}[\|\boldsymbol{X}^{(j)} - \sum_{i:q_i<1} u_i^{(j)}(E_{i*})^{\mathrm{T}}\|_2^2] = \mathbf{E}[\|\boldsymbol{X}^{(j)}\|_2^2] - \|\sum_{i:q_i<1} u_i^{(j)}(E_{i*})^{\mathrm{T}}\|_2^2.$$

Now,

$$\mathbf{E}[\|\boldsymbol{X}^{(j)}\|_2^2] = \mathbf{E}[\|\sum_{i:q_i<1} \frac{u_i^{(j)}}{q_i}(E_{i*})^{\mathrm{T}}\boldsymbol{I}[i \text{ is sampled}]\|_2^2]$$

$$= \sum_{i:q_i<1} \frac{(u_i^{(j)})^2}{q_i^2} \|E_{i*}\|_2^2 q_i + \sum_{i\neq i':q_i,q_{i'}<1} u_i^{(j)} u_{i'}^{(j)} \langle E_{i*}, E_{i'*}\rangle$$

As the values p_i used to define probabilities q_i are such that $p_i \geq \alpha\|E_{i*}\|_2^2/\|E\|_{\mathrm{F}}^2$, then we have

$$\mathbf{E}[\|\boldsymbol{X}^{(j)}\|_2^2] \leq \frac{1}{s\alpha}\|E\|_{\mathrm{F}}^2 + \|\sum_{i:q_i<1} u_i^{(j)}(E_{i*})^{\mathrm{T}}\|_2^2 - \sum_{i:q_i<1}\|u_i^{(j)}(E_{i*})^{\mathrm{T}}\|_2^2.$$

Thus, $\mathbf{E}[\|\boldsymbol{w}^{(j)} - \sigma_j v^{(j)}\|_2^2] \leq (1/s\alpha)\|E\|_{\mathrm{F}}^2 - \sum_{i:q_i<1}\|u_i^{(j)}(E_{i*})^{\mathrm{T}}\|_2^2$. From here, using the same proof as [DRVW06, Theorem 2.1], we obtain that the subspace $V + \mathrm{span}(A_S)$ spans rows of a rank k matrix B such that

$$\|A - B\|_{\mathrm{F}}^2 \leq \|A - A_k\|_{\mathrm{F}}^2 + \frac{k}{s\alpha}\|E\|_{\mathrm{F}}^2. \qquad \square$$

Computing the left factor of an approximation

Let $\boldsymbol{T}$ be a CountSketch matrix with $\Theta(k^2)$ columns. In [CEM+15], the authors show that $\boldsymbol{T}$ is a projection cost preserving sketch, i.e., with probability $9/10$, for all projection matrices P of rank at most $O(k)$,

$$\|(I-P)A\boldsymbol{T}\|_{\mathrm{F}}^2 = (1 \pm 1/10)\|(I-P)A\|_{\mathrm{F}}^2.$$

Let $\boldsymbol{S}$ be a CountSketch matrix with $\Theta(k^4)$ rows. Then, with probability $\geq 99/100$, $\boldsymbol{S}$ is a subspace embedding for the matrix $A\boldsymbol{T}$ and therefore for any matrix X,

$$\|\boldsymbol{S}A\boldsymbol{T}X - \boldsymbol{S}A\boldsymbol{T}\|_{\mathrm{F}}^2 = (1 \pm 1/10)\|A\boldsymbol{T}X - A\boldsymbol{T}\|_{\mathrm{F}}^2.$$

We can relate OPT_A and $\mathrm{OPT}_{\boldsymbol{S}A\boldsymbol{T}}$ as follows:

$$\mathrm{OPT}_{\boldsymbol{S}A\boldsymbol{T}} = \|\boldsymbol{S}A\boldsymbol{T} - [\boldsymbol{S}A\boldsymbol{T}]_k\|_{\mathrm{F}}^2 = \min_{\text{rank-}k\,X}\|\boldsymbol{S}A\boldsymbol{T} - \boldsymbol{S}A\boldsymbol{T}X\|_{\mathrm{F}}^2 \leq \frac{11}{10}\min_{\text{rank-}k\,X}\|A\boldsymbol{T} - A\boldsymbol{T}X\|_{\mathrm{F}}^2 = \frac{11}{10}\mathrm{OPT}_{A\boldsymbol{T}}$$

where the inequality follows from the subspace embedding property of $\boldsymbol{S}$ for the column space of $A\boldsymbol{T}$. Now,

$$\mathrm{OPT}_{A\boldsymbol{T}} = \min_{\text{rank-}k\text{ projections }P}\|(I-P)A\boldsymbol{T}\|_{\mathrm{F}}^2 \leq \frac{10}{9}\min_{\text{rank-}k\text{ projections }P}\|(I-P)A\|_{\mathrm{F}}^2 = \frac{10}{9}\mathrm{OPT}_A.$$

Here, the inequality follows as $\boldsymbol{T}$ is a projection cost preserving sketch for k dimensional projections. Thus, $\mathrm{OPT}_{\boldsymbol{S}A\boldsymbol{T}} \leq (11/9)\mathrm{OPT}_A$.

Boutsidis and Woodruff [BW17] show that for any matrix M, there exists a sub-matrix M' of M, with $O(k/\varepsilon)$ columns such that there is a rank k matrix B, $\mathrm{colspan}(B) \subseteq \mathrm{colspan}(M')$, and

$\|M - B\|_F^2 \le (1+\varepsilon)\|M - [M]_k\|_F^2$. They also give an algorithm to find such a subset of columns. As $\boldsymbol{SAT}$ is an $O(k^4) \times O(k^2)$ matrix, using their algorithm, we can compute in time $\text{poly}(k)$, a column selection matrix Ω that selects $O(k)$ columns of $\boldsymbol{SAT}$ such that

$$\min_{\text{rank-}k\, X} \|\boldsymbol{SAT} - \boldsymbol{SAT}\Omega X\|_F^2 \le \frac{3}{2}\text{OPT}_{SAT} \le 2\text{OPT}_A.$$

We now have $\|(\boldsymbol{SAT}\Omega)(\boldsymbol{SAT}\Omega)^+\boldsymbol{SAT} - \boldsymbol{SAT}\|_F^2 \le \min_{\text{rank-}k\, X} \|\boldsymbol{SAT} - \boldsymbol{SAT}\Omega X\|_F^2 \le 2\text{OPT}_A$. Using the property that $\boldsymbol{S}$ is a subspace embedding for the column space of $A\boldsymbol{T}$, we have

$$\|A\boldsymbol{T}\Omega(\boldsymbol{SAT})^+\boldsymbol{SAT} - A\boldsymbol{T}\|_F^2 \le \frac{20}{11}\text{OPT}_A.$$

Let U be a matrix with orthonormal columns such that $\text{colspan}(A\boldsymbol{T}\Omega) = \text{colspan}(U)$. Therefore,

$$\|UU^T A\boldsymbol{T} - A\boldsymbol{T}\|_F^2 \le \|(A\boldsymbol{T}\Omega)(\boldsymbol{SAT}\Omega)^+\boldsymbol{SAT} - A\boldsymbol{T}\|_F^2 \le \frac{20}{11}\text{OPT}_A$$

which finally implies, as $\boldsymbol{T}$ is a projection cost preserving sketch for $O(k)$ dimensional projections, that $\|UU^T A - A\|_F^2 \le (10/9)(20/11)\text{OPT}_A \le 3\text{OPT}_A$. Thus, $\text{colspan}(U)$ is an $O(k)$ dimensional subspace with $\|(I - UU^T)A\|_F^2 \le 3\text{OPT}_A$. As, $\boldsymbol{T}$ and $\boldsymbol{S}$ are CountSketch matrices, the matrices $A\boldsymbol{T}$ and $\boldsymbol{SAT}$ can be computed in time $\text{nnz}(A)$. The matrix Ω can be computed in time $\text{poly}(k)$ and the matrix $A\boldsymbol{T}\Omega$ is obtained by selecting the appropriate columns of matrix $A\boldsymbol{T}$. The orthonormal matrix U can be computed in time $O(nk^{\omega-1})$. Using U, we now obtain a larger subspace of dimension $O(k/\varepsilon)$ that spans a $1+\varepsilon$ approximation.

Using Lemma 3.5.15, we have that if *columns* of the matrix A are sampled independently to obtain a subset $\boldsymbol{S}_{\text{res}} \subseteq [d]$ such that $\mathbf{Pr}[j \in \boldsymbol{S}_{\text{res}}] \ge \min(1, sp_j)$ for

$$s = O(k/\varepsilon) \text{ and } p_j = \|(I - UU^T)A_{*j}\|_2^2 / \|(I - UU^T)A\|_F^2,$$

then with probability $\ge 99/100$, the subspace $\text{colspan}(U) + \text{colspan}(A^{\boldsymbol{S}_{\text{res}}})$ spans columns of a k dimensional matrix that is a $(1+\varepsilon)$ rank-k approximation for A.

Lemma 3.5.3 shows how to sample $\boldsymbol{S}_{\text{res}}$ from such a distribution. In the notation of Lemma 3.5.3, we have $T_1 = O(\text{nnz}(A) + nk)$ and $T_2 = nk$. Therefore, with probability $\ge 95/100$, we can obtain a sample $\boldsymbol{S}_{\text{res}}$ from a distribution over subsets of $[d]$ such that independently, $\mathbf{Pr}[j \in \boldsymbol{S}_{\text{res}}] \ge \min(1, O(k/\varepsilon)p_j)$ in time $O(\gamma^{-1}(\text{nnz}(A) + nk) + nk\log(d) + \varepsilon^{-1}d^{\gamma}nk\log^2(d)) = O(\gamma^{-1}\,\text{nnz}(A) + \varepsilon^{-1}nkd^{\gamma+o(1)})$ for any small constant γ. Let $M = [U\ A^{\boldsymbol{S}_{\text{res}}}]$. We have with probability $\ge 9/10$, that

$$\min_{\text{rank-}k\, X} \|MX - A\|_F^2 \le (1+\varepsilon)\text{OPT}_A.$$

To obtain a good k-dimensional subspace within the column space of M, we can sketch and solve the above problem. Let $\boldsymbol{T}_1$ be a CountSketch matrix with $O((k/\varepsilon)^2/\varepsilon^2)$ rows. Then with probability

$\geq 99/100$, $\boldsymbol{T}_1$ is an affine embedding for (M, A) and therefore for any matrix X, $\|\boldsymbol{T}_1 MX - \boldsymbol{T}_1 A\|_{\mathrm{F}}^2 \in (1 \pm \varepsilon)\|MX - A\|_{\mathrm{F}}^2$. Let $X_{\boldsymbol{T}_1}$ be the optimal solution for $\min_{\text{rank-}k\, X} \|\boldsymbol{T}_1 MX - \boldsymbol{T}_1 A\|_{\mathrm{F}}$. As $X_{\boldsymbol{T}_1}$ is optimal, the rows of the matrix $X_{\boldsymbol{T}_1}$ must be spanned by the rows of the matrix $\boldsymbol{T}_1 A$, which implies that $\min_{\text{rank-}k\, X} \|MX\boldsymbol{T}_1 A - A\|_{\mathrm{F}}^2 \leq (1 + O(\varepsilon))\mathsf{OPT}_A$. This problem can now be solved by sketching on the left and the right with $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$, where $\boldsymbol{T}_2$ is a CountSketch matrix with $\mathrm{poly}(k/\varepsilon)$ rows, and then solving the sketched problem optimally. The time complexity of sketching is $O(\mathrm{nnz}(M) + \mathrm{nnz}(A)) = O(\mathrm{nnz}(A) + nk/\varepsilon)$, and the sketched problem can be solved in time $\mathrm{poly}(k/\varepsilon)$. Thus in time $O(\mathrm{nnz}(A) + nk/\varepsilon + \mathrm{poly}(k/\varepsilon))$, we can compute a rank k matrix X such that

$$\|MX\boldsymbol{T}_1 A - A\|_{\mathrm{F}}^2 \leq (1 + O(\varepsilon))\mathsf{OPT}_A.$$

We can also compute a decomposition of $X = X_1 \cdot X_2$ where X_1 has k columns in time $\mathrm{poly}(k/\varepsilon)$, which implies that the k dimensional column span of MX_1 is a $1 + O(\varepsilon)$ approximate rank k singular subspace i.e., $\|(MX_1)(MX_1)^+ A - A\|_{\mathrm{F}}^2 \leq (1 + O(\varepsilon))\mathsf{OPT}_A$. The matrix MX_1 can be computed in time $O(nk^{\omega-1}/\varepsilon)$ and a matrix V which is an orthonormal basis for the column space of the $n \times k$ matrix MX_1 can be computed in time $O(nk^{\omega-1})$. Thus, in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + \varepsilon^{-1} nkd^{\gamma+o(1)} + \varepsilon^{-1} nk^{\omega-1} + \mathrm{poly}(\varepsilon^{-1}k))$, we can compute a left factor for a $1 + \varepsilon$ rank-k approximation of A. Thus, we have the following lemma.

Lemma 3.5.16. *Given a matrix $A \in \mathbb{R}^{n \times d}$, a rank parameter k and accuracy parameter ε, we can compute a matrix V with k orthonormal columns in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + \varepsilon^{-1} nkd^{\gamma+o(1)} + \varepsilon^{-(\omega-1)} nk^{\omega-1} + \mathrm{poly}(\varepsilon^{-1}k))$ such that*

$$\|A - VV^{\mathrm{T}}A\|_{\mathrm{F}}^2 \leq (1 + \varepsilon)\|A - [A]_k\|_{\mathrm{F}}^2.$$

Computing a right factor given a left factor

Given a matrix V with k orthonormal columns such that

$$\min_X \|VX - A\|_{\mathrm{F}}^2 \leq (1 + O(\varepsilon))\|A - [A]_k\|_{\mathrm{F}}^2,$$

we want to compute a rank k matrix $\widetilde{X}$ that satisfies $\|V\widetilde{X} - A\|_{\mathrm{F}}^2 \leq (1 + O(\varepsilon))\|A - [A]_k\|_{\mathrm{F}}^2$.

For $i \in [n]$, let $p_i = \|V_{*i}\|_2^2/k$. Suppose $\boldsymbol{S}_{\mathrm{lev}}$ is a sampling matrix with $s = O(k \log(k))$ rows such that each row of $\boldsymbol{S}_{\mathrm{lev}}$ is independently equal to $e_i^{\mathrm{T}}/\sqrt{sp_i}$ with a probability p_i. Then we have

$$\text{for all vectors } x,\ \|\boldsymbol{S}_{\mathrm{lev}} Vx\|_2^2 \in (1 \pm 1/2)\|Vx\|_2^2.$$

Let $M_2 = V^{\mathrm{T}}\boldsymbol{S}_{\mathrm{lev}}^{\mathrm{T}}$ and let V_{M_2} be a matrix with k orthonormal columns such that $\mathrm{colspan}(V_{M_2}) = \mathrm{rowspan}(M_2)$. Let S_2 be the BSS-Sampling matrix returned by the dual set spectral sparsification algorithm of [BW17] on the inputs $V_{M_2}, \boldsymbol{S}_{\mathrm{lev}}(I - VV^{\mathrm{T}})A\boldsymbol{T}$ with a parameter $4k$, where $\boldsymbol{T}$ is a CountSketch matrix with $O(k^2)$ columns. The matrix S_2 selects $4k$ rows of the matrix $\boldsymbol{S}_{\mathrm{lev}}A$. Let $R_1 = S_2\boldsymbol{S}_{\mathrm{lev}}A$.

Lemma 6.7 of [BW17] shows that

$$\|A - AR_1^+ R_1\|_{\mathrm{F}}^2 \le O(1)\|A - [A]_k\|_{\mathrm{F}}^2.$$

As the matrix R_1 has $4k$ rows, an orthonormal basis U for the rowspace of R_1, with $4k$ orthonormal columns, can be computed in time $dk^{\omega-1}$. We can then perform residual sampling of rows of A with respect to the subspace U using the Lemma 3.5.3. Here $T_1 = \mathrm{nnz}(A) + dk$ and $T_2 = dk$. Thus, we can sample rows from a distribution defined by the probabilities

$$\min(1, (s/16)\|A_{i*}(I - UU^{\mathrm{T}})\|_2^2/\|A(I - UU^{\mathrm{T}})\|_{\mathrm{F}}^2),$$

for $s = O(k/\varepsilon)$ in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + \varepsilon^{-1}dkn^{\gamma+o(1)})$. Let $\boldsymbol{S}'_{\mathrm{res}} \subseteq [n]$ be the rows sampled. Let $R = \begin{bmatrix} U^{\mathrm{T}} \\ A_{\boldsymbol{S}'_{\mathrm{res}}} \end{bmatrix}$. The matrix R has $O(k/\varepsilon)$ rows.

Now, as in proof of the Theorem 5.1 of [BW17], we have with probability $\ge 9/10$,

$$\|A - VV^{\mathrm{T}}AR^+R\|_{\mathrm{F}}^2 \le (1 + O(\varepsilon))\|A - [A]_k\|_{\mathrm{F}}^2,$$

which implies $\min_X \|A - VXR\|_{\mathrm{F}}^2 \le (1 + O(\varepsilon))\|A - [A]_k\|_{\mathrm{F}}^2$. By sketching the problem on the left and the right with CountSketch matrices $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ with $\mathrm{poly}(k/\varepsilon)$ rows and columns respectively, the optimal solution $X_{\boldsymbol{T}}$ for the sketched problem satisfies

$$\|A - VX_{\boldsymbol{T}}R\|_{\mathrm{F}}^2 \le (1 + O(\varepsilon))\|A - [A]_k\|_{\mathrm{F}}^2.$$

Finally, the product $X_{\boldsymbol{T}} \cdot R$ can be computed in time $O(dk^{\omega-1}/\varepsilon)$ to obtain a matrix $\widetilde{X}$ such that

$$\|A - V\widetilde{X}\|_{\mathrm{F}}^2 \le (1 + O(\varepsilon))\|A - [A]_k\|_{\mathrm{F}}^2.$$

Thus, we can compute two matrices $V, \widetilde{X}$ with k columns and k rows respectively, such that the product $V \cdot \widetilde{X}$ is a $1 + \varepsilon$ approximate rank-k Frobenius norm approximation to the matrix A, in time

$$O(\gamma^{-1}\,\mathrm{nnz}(A) + \varepsilon^{-1}(n + d)k^{\omega-1} + \varepsilon^{-1}k(nd^{\gamma+o(1)} + dn^{\gamma+o(1)}) + \mathrm{poly}(\varepsilon^{-1}k)).$$

3.6 Conclusions and Open Questions

In this work, we construct the first oblivious subspace embedding with $o(d \log d)$ rows and a distortion $\alpha = \exp(\mathrm{poly}(\log\log d))$ that can be applied to an arbitrary matrix A in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + d^{2+\gamma})$ for any universal constant $\gamma > 0$. This construction leads to first algorithms for problems such as linear regression, independent row computation, etc.

The results in this chapter have been improved by [CSWZ23, CDDR23]. Chenakkod, Dereziński, Dong, and Rudelson [CDDR23] showed that a random matrix $\boldsymbol{S}$ with $O(d)$ rows and polylog(d) nonzero entries per column is an oblivious subspace embedding with a distortion $O(1)$ thus significantly improving upon our construction. Combining this construction with OSNAP, we obtain an oblivious subspace embedding with $O(d)$ rows and a distortion $\alpha = O(1)$ that can be applied to any matrix A in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + d^{2+\gamma})$ for any universal constant $\gamma > 0$.

The main open question is to obtain oblivious subspace embedding constructions with $O(d/\varepsilon^2)$ and a distortion $1 + \varepsilon$ that can be applied to an arbitrary matrix in time $O(\gamma^{-1}\,\mathrm{nnz}(A) + d^{2+\gamma})$ for any constant $\gamma > 0$.

Chapter 4

Dimensionality Reduction for the Sum-of-Distances Objective

4.1 Introduction

Machine learning models often require millions of high-dimensional data samples in order to train. For example, an image with moderate resolution can easily have more than a million pixels. It is crucial that we can decrease the size of the data to save on computational power. One way to decrease the size of the data is dimensionality reduction, where we project our data samples onto a low-dimensional subspace and perform the task on the low-dimensional points. Given a set of n points $A = \{a_1, \ldots, a_n\}$ in $\mathbb{R}^d$, the projections of A onto a subspace P of k dimensions needs only k parameters for each point in the dataset. Thus, the size of the data is proportional to $(n + d)k$, which can be much smaller than nd. Therefore, if there exists a subspace P of dimension k, where k is much smaller than n and d, and for which the projections of A onto the subspace P alone are sufficient to perform a certain a task on the dataset A, then we can achieve a significant reduction in the size of the data.

One very common task that requires dimensionality reduction is the shape-fitting problem. A problem instance is defined by a quadruple $(A, \mathcal{S}, \mathrm{dist}, f)$, where $A = \{ a_1, \ldots, a_n \} \subseteq \mathbb{R}^d$ is a set of points, $\mathrm{dist} : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}_{\geq 0}$ is a metric which we will also refer to as the distance function, $\mathcal{S}$ is a collection of subsets in $\mathbb{R}^d$ which we call *shapes*, and a function $f : \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0}$. The task is to find a shape $S \in \mathcal{S}$ that minimizes $\sum_i f(\mathrm{dist}(a_i, S))$, where $\mathrm{dist}(a_i, S) \coloneqq \inf_{s \in S} \mathrm{dist}(a_i, s)$. The most common function f used is $f(x) = x^2$ as it has a natural Frobenius norm interpretation for many tasks and has closed-form solutions for natural sets $\mathcal{S}$ of shapes. Recently, the function $f(x) = x$ has been considered as it is more robust to outliers than the function $f(x) = x^2$, meaning that it does not square the distance to an erroneous point, allowing the objective to fit more of the remaining (non-outlier) data points.

The most common dimensionality reduction techniques include Principal Component Analysis

(PCA) and the Johnson-Lindenstrauss transform (JL). PCA projects the original dataset onto the space spanned by the top singular vectors. On the other hand, the JL transform provides a data-oblivious dimensionality reduction that preserves pairwise distances between points in the dataset.

Feldman, Schmidt and Sohler [FSS13] show that if P is the subspace spanned by the top $O(k/\varepsilon^2)$ singular vectors of the data matrix A, which is given by PCA, then for any shape S that lies in a k-dimensional space, the quantity $\sum_i \min_{s\in S} \|a_i - s\|_2^2$ can be approximated by $\sum_i \min_{s\in S} \|\mathbb{P}_P a_i - s\|_2^2 + \sum_i \|a_i - \mathbb{P}_P a_i\|_2^2$, where $\mathbb{P}_P a_i$ denotes the Euclidean projection of a_i onto the subspace P, thereby giving a dimensionality reduction technique for the shape-fitting problem instantiated with $f(x) = x^2$, Euclidean norm distance function $\mathrm{dist}(x, y) = \|x - y\|_2$, and with $\mathcal{S}$ being the collection of any k-dimensional shapes.

In this work, we concentrate on shape fitting problems with $\mathrm{dist}(x, y) = \|x - y\|_2$ and $f(x) = x$. Unfortunately, both PCA and the JL transform are not known to work in this case. We give fast algorithms to find a subspace P of $\widetilde{O}(k^3/\varepsilon^6)$ dimensions that allows us to compute a $(1\pm\varepsilon)$-approximation to $\sum_i \mathrm{dist}(x_i, S)$ for any shape S that lies in a k-dimensional subspace. Examples of such shapes include all k-dimensional subspaces themselves, which corresponds to the subspace approximation problem, as well as all sets of k points, which corresponds to the k-median problem. Our results also apply to the (j, l)-projective clustering problem, with $j \cdot l \leq k$, where we seek to find j subspaces, each of dimension at most l, to minimize the sum of distances of each input point to its nearest subspace among the j that we have chosen.

As discussed in Chapter 1, a coreset is another type of data structure to reduce the size of a data set A. Namely, a coreset P is a data structure consuming a much smaller amount of memory than A, which can be used as a substitute for A for any query Y on A. For example, in the k-median problem, the query $Y = \{y_1, \ldots, y_k\}$ can be a set of k points, and we want to find a coreset P to obtain a $(1+\varepsilon)$-approximation to $\sum_{i=1}^n \|a_i - y_{a_i}\|_2$, where y_{a_i} is the closest point to a_i in Y. Often, we want to construct a *strong coreset*, meaning with high probability, P can be used in place of A simultaneously for all possible query sets Y. If this is the case, then we can throw away the original dataset A, which saves us not only on computational power, but also on storage.

There is a long line of work which focuses on constructing coresets for subspace approximation with sum of squared distances loss function, as well as for the k-means problem (see, e.g., [DRVW06, DV07, FL11, FMSW10, FSS13, VX12, SV07, BHPI02, Che09, FS12, FS05, FS08, HPK07, HPM04, LS10]). Feldman, Schmidt and Sohler [FSS13] give the first coresets of size independent of d. For subspace approximation, they give strong coresets of size $O(k/\varepsilon)$, and for the k-means problem, they obtain a coreset of size $\widetilde{O}(k^3/\varepsilon^4)$. [CEM+15] improves the result and give an input sparsity time algorithm to construct the coreset.

Later, Sohler and Woodruff [SW18] give a strong coreset of size $\mathrm{poly}(k/\varepsilon)$ for the k-median problem, as well as the subspace approximation problem with the sum of distances loss function, obtaining the first strong coresets independent of n and d for this problem. Their algorithm runs in $\widetilde{O}(\mathrm{nnz}(A) + (n + d) \cdot \mathrm{poly}(k/\varepsilon) + \exp(\mathrm{poly}(k/\varepsilon)))$ time. Makarychev, Makarychev and Razenshteyn [MMR19] provide an oblivious dimensionality reduction for k-median to an $O(\varepsilon^{-2}\log(k/\varepsilon))$-dimensional space while preserving the cost of every clustering. This dimension reduction result can also be used to construct a strong coreset of size $\mathrm{poly}(k/\varepsilon)$.

Sohler and Woodruff [SW18] gave an algorithm to compute first polynomial size coresets for k-median using their dimensionality reduction, albeit, with a running time exponential in k, $1/\varepsilon$ as discussed. We improve upon their dimensionality reduction algorithm by obtaining an algorithm that does not have the $\exp(\mathrm{poly}(k/\varepsilon))$ term in the running time. We use our dimensionality reduction procedure to obtain an $\widetilde{O}(k^4/\varepsilon^8)$ size coreset for k-median in polynomial time using our dimensionality reduction algorithm. In concurrent and independent work, Huang and Vishnoi [HV20] gave a polynomial time algorithm to compute a coreset of size $\widetilde{O}(k/\varepsilon^4)$. We stress that we can run the second stage in the coreset construction algorithm of [HV20] on a coreset of size $\widetilde{O}(k^4/\varepsilon^8)$ to obtain a coreset of size $\widetilde{O}(k/\varepsilon^4)$ just as in [HV20]. Also, their techniques cannot be extended to give an efficient dimensionality reduction algorithm to approximate the sum-of-distances to an arbitrary k-dimensional shape.

4.1.1 Our Results

Our main contribution is that we obtain the first polynomial time, in fact near-linear time, dimension reduction algorithm that given a matrix A returns a $\mathrm{poly}(k/\varepsilon)$-dimensional subspace such that the projections of the input points to this subspace, as well as the distances of the points to this subspace, can be used to compute a $(1 \pm \varepsilon)$-approximation to the sum of distances of the set A to any k-dimensional shape S.

Theorem 4.1.1 (Dimensionality Reduction). *Given $A \in \mathbb{R}^{n\times d}$ and $0 < \varepsilon < 1$, there exists an algorithm that runs in time $\widetilde{O}(\mathrm{nnz}\,(A)/\varepsilon^2 + (n+d)\,\mathrm{poly}(k/\varepsilon))$ and outputs a subspace P of dimension $\widetilde{O}(k^3/\varepsilon^6)$ such that, with probability $\geq 2/3$, for* any *shape $S \subseteq \mathbb{R}^d$ that lies in a k-dimensional subspace,*

$$\sum_i \sqrt{\mathrm{dist}(\mathbb{P}_P a_i, S)^2 + \mathrm{dist}(a_i, P)^2} = (1 \pm \varepsilon) \sum_i \mathrm{dist}(a_i, S).$$

Given a subspace P as in the above theorem, it is still expensive to compute the projections of the rows of A onto the subspace P as well as the distances to the subspace P. We also give an algorithm to compute approximate projections and approximate distances that still satisfy the guarantees of the above theorems, obtaining the following theorem.

Theorem 4.1.2 (Size Reduction). *Given a matrix $A \in \mathbb{R}^{n\times d}$ and a subspace P of $r = \widetilde{O}(k^3/\varepsilon^6)$ dimensions that satisfies the guarantees of Theorems 4.1.1, there is an algorithm that runs in time $\widetilde{O}(\mathrm{nnz}(A) + (n + d)\,\mathrm{poly}(k/\varepsilon))$ and outputs vectors $a_i^B \in \mathbb{R}^r$ and values $v_i \in \mathbb{R}_{\geq 0}$ for all i such that for any shape S that lies* *in a k dimensional subspace,*

$$\sum_i \sqrt{\mathrm{dist}(Ba_i^B, S)^2 + v_i^2} = (1 \pm \varepsilon) \sum_i \mathrm{dist}(a_i, S),$$

where B is an orthonormal basis for the subspace P. Thus, the storage requirement drops from $\mathrm{nnz}(A)$ *to* $(n+d)k^3/\varepsilon^6$.

4.2 Preliminaries and Technical Overview

We let $A \in \mathbb{R}^{n \times d}$ denote our input matrix. The rows of A are interpreted as a set of n points in $\mathbb{R}^d$. We use A_{i*} and a_i to denote the i^{th} row of A, and A_{*i} to denote the i^{th} column. Similarly, for $J \subseteq [n]$, A_{J*} denotes the matrix with rows of A only indexed by J. For $n \in \mathbb{Z}^+$, $[n]$ denotes the set $\{1, 2, 3, \ldots, n\}$. For a matrix A, we use A^+ to denote its Moore-Penrose pseudoinverse.

Given a subspace B, we use $\mathbb{P}_B$ to denote the projection matrix onto B, i.e., for any vector u, we have $\mathbb{P}_B u = \arg\min_{v \in B} \|u - v\|_2$. Let $B^\perp$ denote the orthogonal complement of the subspace B. We use bold capital letters such as $\boldsymbol{S}, \boldsymbol{L}$ to stress that these are random matrices that are explicitly sampled.

Definition 4.2.1 ($(p,2)$-norm)**.** For a matrix $A \in \mathbb{R}^{n \times d}$, its $(p,2)$-norm is $\|A\|_{p,2} = (\sum_{i=1}^n \|A_{i*}\|_2^p)^{1/p}$. We define $\|A\|_h$ to be $\|A^{\mathrm{T}}\|_{1,2}$ which is the sum of ℓ_2 norms of columns of A.

Definition 4.2.2 ((k,p)-clustering)**.** Given input matrix $A \in \mathbb{R}^{n \times d}$, let $\mathcal{X}$ be the collection of all sets containing k points. The (k,p)-clustering problem denotes the optimization problem

$$\min_{X \in \mathcal{X}} \sum_{A_{i*} \in A} d(A_{i*}, X)^p.$$

If $p = 2$, we have the k-means problem, while if $p = 1$, we have the k-median problem.

Definition 4.2.3 ((k,p)-subspace approximation)**.** Given input matrix $A \in \mathbb{R}^{n \times d}$, let $\mathcal{P}$ be the set of all subspaces with dimension at most k. The (k,p)-subspace approximation problem denotes the optimization problem $\min_{P \in \mathcal{P}} \sum_{i \in [n]} d(A_{i*}, P)^p$. We let $\mathsf{SubApx}_{k,p}(A)$ denote the optimum value of the (k,p) subspace approximation to A.

Definition 4.2.4 (ε-strong coreset)**.** For the (k,p)-clustering problem with input matrix $A \in \mathbb{R}^{n \times d}$, a weighted ε-strong coreset is a tuple (C, w) where $C \in \mathbb{R}^{m \times d}$ and $w : \mathrm{rows}(C) \to \mathbb{R}^+$ is such that simultaneously for all $X \subseteq \mathbb{R}^d$ with $|X| = k$,

$$\sum_{i \in [m]} w(C_{i*}) d(C_{i*}, X)^p = (1 \pm \varepsilon) \sum_{i \in [n]} d(A_{i*}, X)^p.$$

The definition can be generalized to *any* data structure that lets us compute a $(1 \pm \varepsilon)$ approximation to $\sum_{A_{i*} \in A} d(A_{i*}, X)^p$ for all sets X of size k. A similar notion of strong coreset can be defined for the (k, p)-subspace approximation problem as well.

Definition 4.2.5 ((α, β)-bicriteria approximation)**.** Given an input matrix $A \in \mathbb{R}^{n \times d}$ for the $(k, 1)$-subspace approximation problem, we say that a subspace Q is an (α, β)-bicriteria approximation if $\dim(Q) \leq \beta$ and $\sum_{i=1}^{n} d(A_{i*}, Q) \leq \alpha \cdot \text{SubApx}_{k,1}(A)$.

Definition 4.2.6 (ℓ_1 subspace embedding)**.** Let $A \in \mathbb{R}^{n \times d}$, $\Pi \in \mathbb{R}^{s \times n}$. We call Π an (α, β) ℓ_1 subspace embedding if for all $x \in \mathbb{R}^d$, $\alpha \|Ax\|_1 \leq \|\Pi A x\|_1 \leq \beta \|Ax\|_1$. If $QR = \Pi A$ is the QR decomposition, then we let $\|A_{i*} R^{-1}\|_1$ be the ℓ_1 leverage score of the i^{th} row. See [CP15, WW19] for several constructions of ℓ_1 subspace embeddings.

4.2.1 Technical Overview

Let $A \in \mathbb{R}^{n \times d}$ be the input matrix. Sohler and Woodruff [SW18] show that if a subspace S satisfies

$$\|A(I - \mathbb{P}_S)\|_{1,2} - \|A(I - \mathbb{P}_{S+W})\|_{1,2} \leq \varepsilon^2 \cdot \text{SubApx}_{k,1}(A) \tag{4.1}$$

for all k-dimensional subspaces W, then we can reduce the dimension of the input points by projecting the points onto S, while being able to compute a $(1 \pm \varepsilon)$-approximation to the sum of distances to any k-dimensional shape. They construct such a subspace S by directly computing a $(1 + \varepsilon, \text{poly}(k/\varepsilon))$ bicriteria approximation for the $(i^* k, 1)$ subspace approximation problem on A, where i^* is a randomly chosen index in $[1/\varepsilon^2]$. This introduces the $\exp(\text{poly}(k/\varepsilon))$ term in their running time. We show that we can compute $(1 + \varepsilon, \text{poly}(k/\varepsilon))$-bicriteria solutions for the $(k, 1)$-subspace approximation problem on $A(I - P)$, for adaptively chosen projection matrices P, and that with constant probability, the union of the bicriteria solutions we compute has the desired property (4.1).

We solve the problem of finding a $(1 + \varepsilon, \text{poly}(k/\varepsilon))$-bicriteria solution for the $(k, 1)$-subspace approximation problem on the input $A(I - P)$, where P is an arbitrary projection matrix onto a subspace of dimension at most $\text{poly}(k/\varepsilon)$, based on techniques from [CW15]. We simplify their arguments and obtain tighter parameters for their algorithms. We solve the problem in two stages. First we compute an $(O(1), \widetilde{O}(k))$-approximation, i.e., we find a subspace $\hat{X}$ of dimension at most $\widetilde{O}(k)$ such that $\|A(I - P)(I - \mathbb{P}_{\hat{X}})\|_{1,2} \leq O(1) \cdot \text{SubApx}_{k,1}(A(I - P))$.

To achieve this guarantee, we make use of so-called lopsided embeddings. Clarkson and Woodruff [CW15] show that if a matrix S is an ε lopsided embedding for $(V_k, (A(I - P))^{\mathrm{T}})$, where V_k is an orthonormal basis for the k-dimensional subspace that attains the cost $\text{SubApx}_{k,1}(A(I - P))$, then $\min_{\text{rank-}k\ X} \|A(I - P)S^{\mathrm{T}} X - A(I - P)\|_{1,2} \leq (1 + \varepsilon)\text{SubApx}_{k,1}(A(I - P))$. We first show that a Gaussian matrix $\boldsymbol{S}$ with $O(k)$ rows is an $O(1)$ lopsided embedding with probability $\geq 9/10$. Then we show that if a random matrix $\boldsymbol{L}$ is an $O(1)$ ℓ_1 subspace embedding for the matrix $A(I - P)\boldsymbol{S}^{\mathrm{T}}$ and satisfies $\mathbf{E}_{\boldsymbol{L}}[\|\boldsymbol{L}M\|_{1,2}] = \|M\|_{1,2}$ for any fixed matrix M, then the row space of $(\boldsymbol{L}A(I - P))$ is an $O(1)$ approximation. We use the Lewis weight sampling algorithm of Cohen and Peng [CP15] to sample a matrix $\boldsymbol{L}$ that satisfies these properties. As the matrix $\boldsymbol{S}^{\mathrm{T}}$, which is a Gaussian matrix, has only $O(k)$ columns, the matrix $\boldsymbol{L}$ has only $\widetilde{O}(k)$ rows. We can also instead use the ℓ_1 subspace embeddings of Wang and Woodruff [WW19] to construct an $\widetilde{O}(k^{3.5})$-sized ℓ_1 embedding by leverage score sampling [Woo14].

Next, based on the $(O(1), \widetilde{O}(k))$ bicriteria solution, we perform non-adaptive residual sampling. This was shown to give a $(1+\varepsilon, \widetilde{O}(k^3/\varepsilon^2))$ bicriteria solution in [CW15] when an $O(1)$ approximate solution is used. Thus, we obtain a subspace $\hat{S}$ for which

$$\|A(I-P)(I-\mathbb{P}_{\hat{S}})\|_{1,2} \le (1+\varepsilon)\mathrm{SubApx}_{k,1}(A(I-P)).$$

Starting with $P = 0$, we obtain a $(1+\varepsilon,\ k^3/\varepsilon^2)$ bicriteria subspace $\hat{S}$. However, the dimensionality reduction requires a subspace that satisfies (4.1). To obtain such a guarantee, we crucially run this algorithm adaptively $\Theta(1/\varepsilon)$ times. Let $\hat{S}_i$ be the subspace obtained in the i^{th} iteration. In the i^{th} iteration, we find a bicriteria solution for the $(k, 1)$ subspace approximation problem on the matrix $A(I-\mathbb{P}_{\hat{S}_1\cup\ldots\cup\hat{S}_{i-1}})$. We then show that the final subspace $\hat{S} = \cup_j \hat{S}_j$, with probability $\ge 9/10$, satisfies $\|A(I-\mathbb{P}_{\hat{S}})\|_{1,2} - \|A(I-\mathbb{P}_{\hat{S}+W})\|_{1,2} \le \varepsilon\cdot\mathrm{SubApx}_{k,1}(A)$ for all k-dimensional subspaces W. Thus, running the above procedure with parameter ε^2 gives a subspace that satisfies (4.1). We show that each iteration of the algorithm takes $\widetilde{O}(\mathrm{nnz}(A)+(n+d)\,\mathrm{poly}(k/\varepsilon))$ time and as we run the algorithm adaptively for $1/\varepsilon^2$ iterations, the total time complexity of the algorithm is $O(\mathrm{nnz}\,(A)/\varepsilon^2 + (n + d)\,\mathrm{poly}(k/\varepsilon))$.

In addition to providing a tool for data size reduction, our dimensionality reduction also leads to small coreset constructions for various problems with sizes that depend only on the problem parameter k instead of n or d. As shown by [SW18], the points projected onto the subspace given by a dimensionality reduction algorithm can be used to construct coresets of sizes $\mathrm{poly}(k/\varepsilon)$ for k-median and $(k, 1)$-subspace approximation problems. We note that the same constructions work with our dimensionality reduction algorithm.

4.3 Sum of Distances to a k-dimensional shape

Let $A = \{a_1, \ldots, a_n\}$ be a given set of points and P be a $\mathrm{poly}(k/\varepsilon)$ dimensional subspace that satisfies (4.1). Let $S \subseteq \mathbb{R}^d$ be an arbitrary shape such that $\mathrm{span}(S)$ has dimension at most k. We want to obtain an ε approximation to $\sum_i \mathrm{dist}(a_i, S)$.

[SW18] show that for any such shape S,

$$\sum_i \sqrt{\mathrm{dist}(a_i, \mathbb{P}_P a_i)^2 + \mathrm{dist}(\mathbb{P}_P a_i, S)^2} = (1\pm\varepsilon)\sum_{i\in S}\mathrm{dist}(a_i, S).$$

The following lemma is a more general version that works with approximate projections onto the subspace P and approximate distances to the subspace P. A similar lemma is stated as Lemma 14 in

[SW18]. We correct an error in Equation 2 of their proof.

Theorem 4.3.1. *Let P be an r dimensional subspace of $\mathbb{R}^d$ such that*

$$\sum_i \mathrm{dist}(a_i, P) - \sum_i \mathrm{dist}(a_i, P+W) \le \frac{\varepsilon^2}{80}\mathrm{SubApx}_{k,1}(A)$$

for all k-dimensional subspaces W. Let $B \in \mathbb{R}^{d\times r}$ be an orthonormal basis for the subspace P. For each a_i, let $a_i^B \in \mathbb{R}^r$ be such that $\mathrm{dist}(a_i, Ba_i^B) \le (1+\varepsilon_c)\mathrm{dist}(a_i, P)$ and let $(1-\varepsilon_c)\mathrm{dist}(a_i,P) \le \mathrm{apx}_i \le (1+\varepsilon_c)\mathrm{dist}(a_i,P)$ for $\varepsilon_c = \varepsilon^2/6$. Then for any k dimensional shape S, $\sum_i \sqrt{\mathrm{dist}(Ba_i^B, S)^2 + \mathrm{apx}_i^2} = (1 \pm 5\varepsilon)\sum_i \mathrm{dist}(a_i, S)$.

Proof. We have by the Pythagorean theorem that $\mathrm{dist}(Ba_i^B, a_i)^2 = \mathrm{dist}(Ba_i^B, \mathbb{P}_P a_i)^2 + \mathrm{dist}(a_i,P)^2 \le (1+3\varepsilon_c)\mathrm{dist}(a_i,P)^2$ which implies that $\mathrm{dist}(Ba_i^B, \mathbb{P}_P a_i)^2 \le (3\varepsilon_c)\mathrm{dist}(a_i,P)^2$.

Given a shape S, we partition $[n]$ into two sets *small* and *large*. We say an index $i \in [n]$ is *small* if $\mathrm{dist}(\mathbb{P}_P a_i, S) \le \mathrm{dist}(\mathbb{P}_P a_i, Ba_i^B)$. In that case, $\mathrm{dist}(Ba_i^B, S)^2 \le 4\mathrm{dist}(\mathbb{P}_P a_i, Ba_i^B)^2 \le 12\varepsilon_c\mathrm{dist}(a_i,P)^2$ by the triangle inequality and

$$\sqrt{\mathrm{dist}(Ba_i^B, S)^2 + \mathrm{apx}_i^2} \le \sqrt{1+15\varepsilon_c}\,\mathrm{dist}(a_i,P) \le \sqrt{1+15\varepsilon_c}\sqrt{\mathrm{dist}(a_i,P)^2 + \mathrm{dist}(\mathbb{P}_P a_i, S)^2}.$$

Similarly,

$$\sqrt{\mathrm{dist}(Ba_i^B, S)^2 + \mathrm{apx}_i^2} \ge \mathrm{apx}_i \ge (1-\varepsilon_c)\mathrm{dist}(a_i,P) \ge (1-4\varepsilon_c)\sqrt{\mathrm{dist}(\mathbb{P}_P a_i, S)^2 + \mathrm{dist}(a_i,P)^2}$$

by using the fact that $\mathrm{dist}(\mathbb{P}_P a_i, S)^2 \le \mathrm{dist}(\mathbb{P}_P a_i, Ba_i^B)^2 \le 3\varepsilon_c\mathrm{dist}(a_i,P)^2$.

We say that any $i \in [n]$ that is not *small*, is *large*. By the triangle inequality, we obtain that

$$\mathrm{dist}(\mathbb{P}_P a_i, S) - \mathrm{dist}(\mathbb{P}_P a_i, Ba_i^B) \le \mathrm{dist}(Ba_i^B, S) \le \mathrm{dist}(\mathbb{P}_P a_i, S) + \mathrm{dist}(Ba_i^B, \mathbb{P}_P a_i). \tag{4.2}$$

As i is *large*, $\mathrm{dist}(\mathbb{P}_P a_i, S) - \mathrm{dist}(\mathbb{P}_P a_i, Ba_i^B) > 0$ and therefore by the AM-GM inequality, we obtain that

$$\mathrm{dist}(Ba_i^B, S)^2 = (1\pm\varepsilon)\mathrm{dist}(\mathbb{P}_P a_i, S)^2 + \left(1 \pm \frac{1}{\varepsilon}\right)\mathrm{dist}(Ba_i^B, \mathbb{P}_P a_i)^2.$$

Thus,

$$\mathrm{dist}(Ba_i^B, S)^2 \le (1+\varepsilon)\mathrm{dist}(\mathbb{P}_P a_i, S)^2 + (2/\varepsilon)(3\varepsilon_c)\mathrm{dist}(a_i,P)^2 \qquad \text{and}$$
$$\mathrm{dist}(Ba_i^B, S)^2 \ge (1-\varepsilon)\mathrm{dist}(\mathbb{P}_P a_i, S)^2 - (1/\varepsilon)(3\varepsilon_c)\mathrm{dist}(a_i,P)^2.$$

Letting $\varepsilon_c = \varepsilon^2/6$, we finally have

$$\mathrm{dist}(Ba_i^B, S)^2 + \mathrm{apx}_i^2 \le (1+\varepsilon)\mathrm{dist}(\mathbb{P}_P a_i, S)^2 + (1+2\varepsilon)\mathrm{dist}(a_i,P)^2$$

and

$$\mathrm{dist}(Ba_i^B, S)^2 + \mathrm{apx}_i^2 \geq (1-\varepsilon)\mathrm{dist}(\mathbb{P}_P a_i, S)^2 + (1-3\varepsilon)\mathrm{dist}(a_i, P)^2.$$

Therefore, by combining both *small* and *large* indices,

$$\sum_i \sqrt{\mathrm{dist}(Ba_i^B, S)^2 + \mathrm{apx}_i^2} \leq \sqrt{1+O(\varepsilon)} \sum_i \sqrt{\mathrm{dist}(\mathbb{P}_P a_i, S)^2 + \mathrm{dist}(a_i, P)^2}$$

and

$$\sum_i \sqrt{\mathrm{dist}(Ba_i^B, S)^2 + \mathrm{apx}_i^2} \geq \sqrt{1-O(\varepsilon)} \sum_i \sqrt{\mathrm{dist}(\mathbb{P}_P a_i, S)^2 + \mathrm{dist}(a_i, P)^2}.$$

The theorem now follows from Theorem 8 of [SW18]. □

The above theorem shows that we have to only compute approximate projections onto the subspace, which can be done in input sparsity time by using high probability subspace embeddings obtained from CountSketch matrices (see Section 2.3 of [Woo14] and [LBKW14]).

4.4 Dimensionality Reduction

4.4.1 Constructing an $(O(1), \widetilde{O}(k))$-bicriteria Subspace Approximation

We first show how to obtain an $(O(1), \widetilde{O}(k))$-bicriteria solution for $(k, 1)$-subspace approximation. A key tool we use is a lopsided embedding defined as follows:

Definition 4.4.1 (Lopsided embedding). A matrix S is a lopsided ε-embedding for matrices A and B with respect to a matrix norm $\|\cdot\|$ and constraint set $\mathcal{C}$, if (i) for all matrices X of the appropriate dimensions, $\|S(AX-B)\| \geq (1-\varepsilon)\|AX-B\|$, and (ii) for $B^* = AX^* - B$, we have $\|SB^*\| \leq (1+\varepsilon)\|B^*\|$, where $X^* = \arg\min_{X\in\mathcal{C}} \|AX - B\|$.

Let $U_k \in \mathbb{R}^{n\times k}$ and $V_k^{\mathrm{T}} \in \mathbb{R}^{k\times d}$ be rank k matrices such that $\|U_k V_k^{\mathrm{T}} - A\|_{1,2} = \mathrm{SubApx}_{k,1}(A)$. Clarkson and Woodruff [CW15] show that if S is a lopsided ε-embedding for matrices (V_k, A^{T}) with respect to the norm $\|\cdot\|_h$, then $\min_{\text{rank-}k\ X} \|AS^{\mathrm{T}}X - A\|_{1,2} \leq (1+O(\varepsilon))\mathrm{SubApx}_{k,1}(A)$. We show that a suitably scaled Gaussian random matrix $\boldsymbol{S}$ with $\widetilde{O}(k)$ rows is a lopsided $(1/4)$-embedding for matrices (V_k, A^{T}) with probability $\geq 9/10$. Thus, we have that with probability $\geq 9/10$,

$$\min_{\text{rank-}k\ X} \|AS^{\mathrm{T}}X - A\|_{1,2} \leq (3/2)\mathrm{SubApx}_{k,1}(A).$$

We next prove that a row-sampling based ℓ_1 subspace embedding for the column space of the matrix AS^{T} can be used to obtain a bicriteria solution to the subspace approximation problem.

The following lemma summarizes the results discussed above. The results of the lemma are a significant improvement over Lemma 44 of [CW15] and have simpler proofs that do not involve ε-nets.

Lemma 4.4.2. *(i) If $\boldsymbol{S}^{\mathrm{T}}$ is a random Gaussian matrix with $O(k)$ columns, then $\boldsymbol{S}$ is a $1/4$-lopsided embedding for (V_k, A^{T}) with respect to the $\|\cdot\|_h$ norm with probability $\geq 9/10$. Therefore, with probability $\geq 9/10$*

$$\min_{\text{rank-}k\ X} \|A\boldsymbol{S}^{\mathrm{T}}X - A\|_{1,2} \leq (3/2)\mathrm{SubApx}_{k,1}(A).$$

(ii) If $\boldsymbol{L}$ is a random matrix drawn from a distribution such that with probability $\geq 9/10$, $\alpha\|A\boldsymbol{S}^{\mathrm{T}}y\|_1 \leq \|\boldsymbol{L}A\boldsymbol{S}^{\mathrm{T}}y\|_1 \leq \beta\|A\boldsymbol{S}^{\mathrm{T}}y\|_1$ for all vectors y and if $\mathbf{E}_{\boldsymbol{L}}[\|\boldsymbol{L}M\|_{1,2}] = \|M\|_{1,2}$ for any matrix M, then with probability $\geq 3/5$, all matrices X of appropriate dimensions such that $\|\boldsymbol{L}A\boldsymbol{S}^{\mathrm{T}}X - \boldsymbol{L}A\|_{1,2} \leq 10 \cdot \mathrm{SubApx}_{k,1}(A)$ satisfy $\|A\boldsymbol{S}^{\mathrm{T}}X - A\|_{1,2} \leq O(2 + 40/\alpha) \cdot \mathrm{SubApx}_{k,1}(A)$.

We defer the proof of this lemma to the appendix as it mostly uses existing proof ideas. Using the above lemma, we now have the following theorem which shows that Algorithm 4.1 returns an $(O(1), \widetilde{O}(k))$ approximation.

Algorithm 4.1: PolyApprox

```
    Input: $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{d\times c_1}$, $k \in \mathbb{Z}$, $\delta$
    Output: $\hat{X} \in \mathbb{R}^{d\times c_2}$
1   cols $\leftarrow O(k + 1/\delta^2)$
2   $\boldsymbol{S}^{\mathrm{T}} \leftarrow \mathcal{N}(0,1)^{d\times \mathrm{cols}}$
3   $\boldsymbol{L} \leftarrow$ LewisWeight$(A(I - BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}, 1/2)$ [CP15]
4   $\hat{X} \leftarrow$ Orthonormal Basis for rowspace$(\boldsymbol{L}A(I - BB^{\mathrm{T}}))$
5   Repeat the above $O(\log(1/\delta))$ times and return the best $\hat{X}$ i.e., $\hat{X}$ minimizing
      $\|A(I - \hat{X}\hat{X}^{\mathrm{T}})\boldsymbol{G}\|_{1,2}$ where $\boldsymbol{G}$ is a Gaussian matrix with $O(\log(n))$ columns
```

Theorem 4.4.3. *Given any matrix $A \in \mathbb{R}^{n\times d}$ and a matrix $B \in \mathbb{R}^{d\times c_1}$ with $c_1 = \mathrm{poly}(k/\varepsilon)$ orthonormal columns, Algorithm 4.1 returns a matrix $\hat{X}$ with $\widetilde{O}(k)$ orthonormal columns that with probability $1-\delta$ satisfies*

$$\|A(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_{1,2} \leq O(1) \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})),$$

in time $\widetilde{O}((\mathrm{nnz}(A) + d\,\mathrm{poly}(k/\varepsilon))\log(1/\delta))$.

Proof. From Lemma 4.4.2, It is shown in Lemma A.1.3, since $\boldsymbol{S}^{\mathrm{T}}$ is a Gaussian matrix with $O(k)$ columns, we obtain that

$$\min_{\text{rank-}k\ X} \|A(I - BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}X - A(I - BB^{\mathrm{T}})\|_{1,2} \leq (3/2)\mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}}))$$

with probability $\geq 9/10$. [CP15] show that a sampling matrix $\boldsymbol{L}$ obtained using Lewis weights has $\widetilde{O}(k)$ rows and is a $(1/2, 3/2)$ ℓ_1 subspace embedding for the matrix $A(I - BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}$. Thus, the matrices $\boldsymbol{S}^{\mathrm{T}}$ and $\boldsymbol{L}$ constructed in Algorithm 4.1 satisfy the conditions of Lemma 4.4.2. Therefore, with probability $\geq 3/5$, if a matrix X satisfies $\|\boldsymbol{L}A(I-BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}X - \boldsymbol{L}A(I-BB^{\mathrm{T}})\|_{1,2} \leq 10\cdot\mathrm{SubApx}_{k,1}(A(I-BB^{\mathrm{T}}))$, then $\|A(I - BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}X - A(I - BB^{\mathrm{T}})\|_{1,2} \leq 82 \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}}))$.

Let $\widetilde{X} = \arg\min_{\text{rank-}k\ X} \|A(I - BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}X - A(I - BB^{\mathrm{T}})\|_{1,2}$. We have

$$\|A(I - BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}\widetilde{X} - A(I - BB^{\mathrm{T}})\|_{1,2} \le (3/2)\mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})).$$

By Markov's bound, with probability $\ge 3/4$,

$$\|\boldsymbol{L}A(I - BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}\widetilde{X} - \boldsymbol{L}A(I - BB^{\mathrm{T}})\|_{1,2} \le 10 \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})).$$

We now have the following:

$$\|\boldsymbol{L}A(I-BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}\widetilde{X}(\boldsymbol{L}A(I-BB^{\mathrm{T}}))^{+}\boldsymbol{L}A(I-BB^{\mathrm{T}}) - \boldsymbol{L}A(I-BB^{\mathrm{T}})\|_{1,2} \le 10\cdot\mathrm{SubApx}_{k,1}(A(I-BB^{\mathrm{T}})).$$

Thus, $\|A(I-BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}\widetilde{X}(\boldsymbol{L}A(I-BB^{\mathrm{T}}))^{+}\boldsymbol{L}A(I-BB^{\mathrm{T}})-A(I-BB^{\mathrm{T}})\|_{1,2} \le 82\cdot\mathrm{SubApx}_{k,1}(A(I-BB^{\mathrm{T}}))$. Finally,

$$\begin{aligned}&\|A(I - BB^{\mathrm{T}})(\boldsymbol{L}A(I - BB^{\mathrm{T}}))^{+}(\boldsymbol{L}A(I - BB^{\mathrm{T}})) - A(I - BB^{\mathrm{T}})\|_{1,2}\\ &\le \|A(I - BB^{\mathrm{T}})\boldsymbol{S}^{\mathrm{T}}\widetilde{X}(\boldsymbol{L}A(I - BB^{\mathrm{T}}))^{+}\boldsymbol{L}A(I - BB^{\mathrm{T}}) - A(I - BB^{\mathrm{T}})\|_{1,2}\\ &\le 82 \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})).\end{aligned}$$

Here we use the fact that for all x and y, $\|x^{\mathrm{T}}(\boldsymbol{L}A)^{+}(\boldsymbol{L}A) - x^{\mathrm{T}}\|_2 \le \|y^{\mathrm{T}}(\boldsymbol{L}A)^{+}(\boldsymbol{L}A) - x^{\mathrm{T}}\|_2$.

By a union bound, with probability $\ge 1/2$, the matrix $\hat{X}$ computed by Algorithm 4.1, which is an orthonormal basis for the rowspace of $\boldsymbol{L}A(I - BB^{\mathrm{T}})$, satisfies

$$\|A(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_{1,2} \le 82 \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})).$$

Thus, the matrix $\hat{X}$ which has the minimum value over $\widetilde{O}(\log(1/\delta))$ trials satisfies with probability $\ge 1 - \delta$ that

$$\|A(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_{1,2} \le O(1) \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})).$$

The running time of Lewis weight sampling can be seen to be $O((\mathrm{nnz}(A)+k^2d(c_1+k))\log(\log(n)))$ from [CP15]. Thus, the total running time is $\widetilde{O}((\mathrm{nnz}(A) + k^2d(c_1 + k))\log(1/\delta))$. □

4.4.2 Constructing a $(1 + \varepsilon, \widetilde{O}(k^3/\varepsilon^2))$-bicriteria Subspace Approximation

Using the $(O(1), \widetilde{O}(k))$-bicriteria subspace approximation solution found, we design a finer sampling process based on Theorem 45 of [CW15] to further pick a subspace of dimension $\widetilde{O}(k^3/\varepsilon^2)$ that contains a $(1 + \varepsilon)$-approximate solution for subspace approximation of the matrix $A(I - BB^{\mathrm{T}})$.

The following lemma states that given a subspace of cost at most $K \cdot \mathrm{SubApx}_{k,1}(A)$, that a sample of $\widetilde{O}(K\cdot k^3/\varepsilon^2)$ rows with probabilities chosen proportional to the distances of the rows of the matrix A to the subspace, can be used to construct a subspace that is a $1 + \varepsilon$ approximation.

Algorithm 4.2: EPSAPPROX

```
Input: $A, B, \hat{X}, k, K, \varepsilon, \delta > 0$.
Output: $U \in \mathbb{R}^{d\times c}$ such that $U^{\mathrm{T}}B = 0$
1 $t \leftarrow O(\log(n/\delta)), \boldsymbol{G} \leftarrow \mathcal{N}(0, 1/t)^{d\times t}$
2 $M \leftarrow A(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\boldsymbol{G}$
3 $p_i \leftarrow \|M_{i*}\|_2/\|M\|_{1,2}$ for all $i \in [n]$
4 $s \leftarrow \widetilde{O}(K \cdot k^3/\varepsilon^2 \cdot \log(1/\delta))$
5 $\boldsymbol{S} \leftarrow$ Multiset of $s$ independent samples drawn from distribution $p$
6 $U \leftarrow$ Orthonormal basis for column space of the matrix $((I - BB^{\mathrm{T}})[\hat{X}\ (A_{\boldsymbol{S}})^{\mathrm{T}}])$ return $U$
```

Lemma 4.4.4. *Given a matrix $A \in \mathbb{R}^{n\times d}$ and a matrix $\hat{X} \in \mathbb{R}^{d\times c}$ that satisfies*

$$\|A(I - \hat{X}\hat{X}^{\mathrm{T}})\|_{1,2} \le K \cdot \mathrm{SubApx}_{k,1}(A),$$

suppose we generate a matrix $\boldsymbol{S}$ of $s = \widetilde{O}((K/\alpha) \cdot k^3/\varepsilon^2 \cdot \log(1/\delta))$ rows, each chosen independently to be the i^{th} standard basis vector with probability p_i. Here, $\sum_{i\in[n]} p_i = 1$ and for all $i \in [n]$ $p_i \ge \alpha\frac{q_i}{\sum_i q_i}$, where $q_i = \|A_{i*}(I - \hat{X}\hat{X}^{\mathrm{T}})\|_2$. Let U be an orthonormal basis for the rowspace of $[\hat{X}^{\mathrm{T}}\ ;\ \boldsymbol{S}A]$. Then with probability $\ge 1 - \delta$,

$$\|A(I - UU^{\mathrm{T}})\|_{1,2} \le (1 + \varepsilon)\mathrm{SubApx}_{k,1}(A).$$

The proof of the above lemma is the same as that of the proof of Theorem 45 of [CW15] with a minor change to account for the approximation error α. Now the following theorem shows that Algorithm 4.2 satisfies conditions of the previous lemma.

Theorem 4.4.5 (Residual Sampling). *Given matrix $A \in \mathbb{R}^{n\times d}$, matrices $B \in \mathbb{R}^{d\times c_1}$ and $\hat{X} \in \mathbb{R}^{d\times c_2}$ with orthonormal columns such that $\|A(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_{1,2} \le K \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}}))$, Algorithm 4.2 returns a matrix U having $c = \widetilde{O}(c_2 + K \cdot k^3/\varepsilon^2 \cdot \log(1/\delta))$ orthonormal columns such that with probability $\ge 1 - \delta$,*

$$\begin{aligned}&\|A(I - BB^{\mathrm{T}})(I - UU^{\mathrm{T}})\|_{1,2}\\ &\le (1 + \varepsilon)\mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})).\end{aligned}$$

The algorithm runs in time $\widetilde{O}(\mathrm{nnz}(A) + d\,\mathrm{poly}(k/\varepsilon))$. Moreover, we also have that $U^{\mathrm{T}}B = 0$ i.e., the column spaces of U and B are orthogonal to each other.

Proof. As the matrix G is a Gaussian matrix with $t = O(\log(n/\delta))$ columns, we have that with probability $\ge 1 - (\delta/2)$, for all $i \in [n]$,

$$\|M_{i*}\|_2 = \|A_{i*}(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})G\|_2 = (1 \pm 1/10)\|A_{i*}(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})G\|_2.$$

Therefore, the probabilities p_i computed by Algorithm 4.2 are such that

$$p_i = \frac{\|M_{i*}\|_2}{\|M\|_{1,2}} \geq \frac{(9/10)\|A_{i*}(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_2}{(11/10)\|A(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_{1,2}} \geq \frac{9}{11}\frac{\|A_{i*}(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_2}{\|A(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_{1,2}}.$$

Hence, by applying Lemma 4.4.4 to the matrix $A(I - BB^{\mathrm{T}})$, we obtain that with probability $\geq 1 - \delta$, the matrix U returned by Algorithm 4.2 satisfies

$$\|A(I - BB^{\mathrm{T}})(I - UU^{\mathrm{T}})\|_{1,2} \leq (1 + \varepsilon)\mathrm{SubApx}_{1,k}(A(I - BB^{\mathrm{T}})).$$

The matrix M can be computed in time $O(\mathrm{nnz}(A)\log(n/\delta) + (c_1 + c_2)d\log(n/\delta))$. And $s = \widetilde{O}(K \cdot k^3/\varepsilon^2 \cdot \log(1/\delta))$ independent samples can be drawn from the distribution p in time $O(n+s)$. Finally, the orthonormal basis U can be computed in time $O(d(c + c_1)^2) = O(d\,\mathrm{poly}(k/\varepsilon))$. □

Therefore, using the $(O(1), \widetilde{O}(k))$ bicriteria solution obtained using Algorithm 4.1, we can obtain a $(1 + \varepsilon, \widetilde{O}(k^3/\varepsilon^2))$ bicriteria solution.

4.5 Dimensionality Reduction

With an algorithm to construct a $(1 + \varepsilon, k^3/\varepsilon^2)$ bicriteria solution from the previous section, we are now ready to construct a subspace that satisfies (4.1). Recall the crucial property for the subspace S we need is that for all k-dimensional subspaces W, $\|A(I - \mathbb{P}_S)\|_{1,2} - \|A(I - \mathbb{P}_{S+W})\|_{1,2} \leq \varepsilon^2\mathrm{SubApx}_{k,1}(A)$. To get such a subspace, we run Algorithms 4.1 and 4.2 adaptively and then show that the union of all $1 + \varepsilon$ approximate bicriteria solutions satisfy the above property with parameter $O(\varepsilon)$.

Lemma 4.5.1. *With probability $\geq 2/3$, Algorithm 4.3 finds an $\widetilde{O}(k^3/\varepsilon^3)$-dimensional subspace S such that for all k-dimensional subspaces W,*

$$\|A(I - \mathbb{P}_S)\|_{1,2} - \|A(I - \mathbb{P}_{S+W})\|_{1,2} \leq 4\varepsilon \cdot \mathrm{SubApx}_{k,1}(A).$$

Proof. Suppose that the loop in Algorithm 4.3 is run for all $t = 10/\varepsilon + 1$ iterations instead of stopping after i^* iterations. Let $\hat{X}_i, U_i, B_i$ be the values of the matrices in the algorithm at the end of i iterations. Let $B_0 = []$ be the empty matrix. Condition on the event that all the calls to Algorithm 4.1 in the algorithm succeed. By a union bound over the failure event of each call to Algorithm 4.1, this event holds with probability $\geq 9/10$. Therefore, by Theorem 4.4.3, we obtain that

$$\|A(I - \mathbb{P}_{B_{i-1}})(I - \mathbb{P}_{\hat{X}_i})\|_{1,2} \leq \widetilde{O}(\sqrt{k}) \cdot \mathrm{SubApx}_{k,1}(A(I - \mathbb{P}_{B_{i-1}}))$$

for all $i \in [10/\varepsilon + 1]$ and also that $\hat{X}_i$ has $\widetilde{O}(k)$ columns. Now we condition on the event that all the calls to Algorithm 4.2 succeed. By a union bound, this holds with probability $\geq 9/10$. Thus, we have

$$\|A(I - \mathbb{P}_{B_i})\|_{1,2} = \|A(I - \mathbb{P}_{B_{i-1}})(I - \mathbb{P}_{U_i})\|_{1,2} \leq (1+\varepsilon) \cdot \mathsf{SubApx}_{k,1}(A(I - \mathbb{P}_{B_{i-1}}))$$

for all iterations $i \in [10/\varepsilon + 1]$ and also that U_i has $\widetilde{O}(k^3/\varepsilon^2)$ columns which implies that B_i has $\widetilde{O}(ik^3/\varepsilon^2)$ columns. In particular, we have that $\|A(I - \mathbb{P}_{B_1})\|_{1,2} \leq (1+\varepsilon)\mathsf{SubApx}_{k,1}(A)$. Therefore,

$$\begin{aligned}(1+\varepsilon)\mathsf{SubApx}_{k,1}(A) - \|A(I - \mathbb{P}_{B_t})\|_{1,2} &\geq \|A(I - \mathbb{P}_{B_1})\|_{1,2} - \|A(I - \mathbb{P}_{B_t})\|_{1,2} \\ &= \sum_{i=2}^{\mathrm{T}} \|A(I - \mathbb{P}_{B_{i-1}})\|_{1,2} - \|A(I - \mathbb{P}_{B_i})\|_{1,2} \geq 0.\end{aligned}$$

The last inequality follows from the fact that $\mathrm{colspace}(B_i) \supseteq \mathrm{colspace}(B_{i-1})$. The summation in the above equation has $10/\varepsilon$ non-negative summands that all sum to at most $(1+\varepsilon)\mathsf{SubApx}_{k,1}(A)$. Therefore, at least $9/\varepsilon$ summands have value $\leq \varepsilon(1+\varepsilon)\mathsf{SubApx}_{k,1}(A)$. In particular, with probability $\geq 9/10$,

$$\|A(I - \mathbb{P}_{B_{i^*}})\|_{1,2} - \|A(I - \mathbb{P}_{B_{i^*+1}})\|_{1,2} \leq \varepsilon(1+\varepsilon)\mathsf{SubApx}_{k,1}(A).$$

But we also have that

$$\begin{aligned}\|A(I - \mathbb{P}_{B_{i^*+1}})\|_{1,2} &= \|A(I - \mathbb{P}_{B_{i^*}})(I - \mathbb{P}_{U_{i^*}})\|_{1,2} \\ &\leq (1+\varepsilon)\mathsf{SubApx}_{k,1}(A(I - \mathbb{P}_{B_{i^*}})) \\ &\leq (1+\varepsilon)\|A(I - \mathbb{P}_{B_{i^*}})(I - \mathbb{P}_W)\|_{1,2} \\ &= (1+\varepsilon)\|A(I - \mathbb{P}_{B_{i^*}+W})\|_{1,2}\end{aligned}$$

where W is any rank k matrix. The second inequality follows from the fact that $\mathsf{SubApx}_{k,1}(A(I - \mathbb{P}_{B_{i^*}})) = \min_{\text{rank-}k\ W} \|A(I - \mathbb{P}_{B_{i^*}})(I - \mathbb{P}_W)\|_{1,2}$. Therefore, for any rank-k matrix W, we obtain that

$$\begin{aligned}&\|A(I - \mathbb{P}_{B_{i^*}})\|_{1,2} - \|A(I - \mathbb{P}_{B_{i^*} \cup W})\|_{1,2} \\ &\leq \|A(I - \mathbb{P}_{B_{i^*}})\|_{1,2} - \frac{1}{1+\varepsilon}\|A(I - \mathbb{P}_{B_{i^*+1}})\|_{1,2} \\ &\leq \|A(I - \mathbb{P}_{B_{i^*}})\|_{1,2} - (1-\varepsilon)\|A(I - \mathbb{P}_{B_{i^*+1}})\|_{1,2} \\ &\leq (\|A(I - \mathbb{P}_{B_{i^*}})\|_{1,2} - \|A(I - \mathbb{P}_{B_{i^*+1}})\|_{1,2}) + \varepsilon\|A(I - \mathbb{P}_{B_{i^*+1}})\|_{1,2} \\ &\leq 4\varepsilon \cdot \mathsf{SubApx}_{k,1}(A).\end{aligned}$$

□

From the above lemma we have that running the algorithm with parameter $\Theta(\varepsilon^2)$ gives a subspace with the desired property and lets us obtain our main theorem.

Theorem 4.5.2. *Given a matrix $A \in \mathbb{R}^{n \times d}$, $k \in \mathbb{Z}$ and an accuracy parameter $\varepsilon > 0$, Algorithm 4.4 returns a matrix B with $\widetilde{O}(k^3/\varepsilon^6)$ orthonormal columns and a matrix $\mathsf{Apx} = [X\ v]$ such that for any k dimensional* *shape S, $\sum_i \sqrt{\mathrm{dist}(BX_{i*}^{\mathrm{T}}, S)^2 + v_i^2} = (1 \pm \varepsilon) \sum_i \mathrm{dist}(A_i, S)$. The algorithm runs in time $O(\mathrm{nnz}\,(A)/\varepsilon^2 + (n+d)\,\mathrm{poly}(k/\varepsilon))$.*

Proof. From the above lemma, we have that the subspace B satisfies with probability $\geq 9/10$, that for any k dimensional subspace W,

$$\|A(I - \mathbb{P}_B)\|_{1,2} - \|A(I - \mathbb{P}_{B \cup W})\|_{1,2} \leq \frac{\varepsilon^2}{80} \mathrm{SubApx}_{k,1}(A). \tag{4.3}$$

From Theorem 2.10 of [Woo14], we obtain that with probability $\geq 9/10$, for all $i \in [n]$, the matrix S_j found for $i \in [n]$ is such that S_j is a $\Theta(\varepsilon^2)$ subspace embedding for the matrix $[B\, A_{i*}^{\mathrm{T}}]$. Therefore, x_i is such that

$$\|Bx_i - A_{i*}^{\mathrm{T}}\|_2 \leq (1 + \Theta(\varepsilon^2))\|(I - BB^{\mathrm{T}})A_{i*}^{\mathrm{T}}\|_2,$$

and $v_i = (1 \pm \Theta(\varepsilon^2))\|(I - BB^{\mathrm{T}})A_{i*}^{\mathrm{T}}\|_2$. Now the proof follows from Theorem 4.3.1. □

Algorithm 4.3: DimensionReduction

```
Input: $A \in \mathbb{R}^{n \times d}$, $k, \varepsilon > 0$.
Output: $B \in \mathbb{R}^{d \times c}$ with orthonormal columns
1 $i^* \leftarrow$ uniform random integer from $[10/\varepsilon + 1]$.
2 Initialize $B \leftarrow []$
3 for $i^*$ iterations do
4     $\hat{X} \leftarrow \textsc{PolyApprox}(A, B, k, \varepsilon/100)$.
5     $U \leftarrow \textsc{EpsApprox}(A, B, \hat{X}, k, \widetilde{O}(\sqrt{k}), \varepsilon, \varepsilon/100)$
6     $B \leftarrow [B \,|\, U]$.
7 end
8 return $B$
```

4.6 Coreset Construction using Dimensionality Reduction

Algorithm 4.5 gives the general algorithm to construct a coreset for any objective involving the sum-of-distances metric. In this section, we discuss the coreset construction for two such problems: the k-median and k-subspace approximation problems.

For $(B, \mathrm{Apx} = [X\, v])$ returned by Algorithm 4.4, we have the guarantee that, with probability $\geq 9/10$, for any k-dimensional shape S,

$$\sum_i \sqrt{\mathrm{dist}(BX_{i*}^{\mathrm{T}}, S)^2 + v_i^2} = (1 \pm \varepsilon) \sum_i \mathrm{dist}(A_{i*}, S).$$

Algorithm 4.4: CompleteDimReduce

```
   Input: $A \in \mathbb{R}^{n \times d}, k \in \mathbb{Z}, \varepsilon > 0$.
   Output: Apx $\in \mathbb{R}^{n \times (c+1)}$
1  Let $B = \textsc{DimensionReduction}(A, k, \Theta(\varepsilon^2))$.
2  $t = O(\log(n))$
3  Compute $(S_j B, S_j A^{\mathrm{T}})$ for $j \in [t]$ where $S_j$ is an independent CountSketch matrix with
     poly$(k/\varepsilon)$ rows
4  for $i = 1, \dots, n$ do
5      Let $U_j D_j V_j^{\mathrm{T}} \leftarrow \mathrm{SVD}(S_j [B\, A_{i*}^{\mathrm{T}}])$ for all $j \in [t]$
6      for $j \in [t]$ do
7          Check if for at least half $j' \neq j$, all singular values of $D_j V_j^{\mathrm{T}} V_{j'} (D_{j'}^{\mathrm{T}})^{-1}$ are in
             $[1 - \Theta(\varepsilon^2), 1 + \Theta(\varepsilon^2)]$
8          If the above check holds, set
               $x_i \leftarrow (S_j B)^{\dagger} (S_j A_{i*}^{\mathrm{T}})$
               $v_i \leftarrow \|(I - (S_j B)(S_j B)^{\dagger})(S_j A_{i*}^{\mathrm{T}})\|_2$
            and go to next $i$
9      end
10 end
11 return $B$ and $n \times (c+1)$ matrix Apx with $\mathrm{Apx}_{i*} = [x_i \; v_i]$
```

Algorithm 4.5: CoresetConstruction

```
   Input: $A \in \mathbb{R}^{n \times d}$, $k$, $\varepsilon$
   Output: Coreset
1  $(B, \mathrm{Apx}) \leftarrow \textsc{CompleteDimReduce}(A, k, \varepsilon)$
2  Construct a coreset for the instance Apx $\begin{bmatrix} B^{\mathrm{T}} & 0 \\ 0 & 1 \end{bmatrix}$ and return
```

Given a set S, let S_{+1} denote the set $\{(s,0) \mid s \in S\}$. Let $\mathrm{diag}(B^{\mathrm{T}},1) = \begin{bmatrix} B^{\mathrm{T}} & 0 \\ 0 & 1 \end{bmatrix}$. Using this notation, we have that

$$\sum_i \mathrm{dist}(\mathrm{Apx}_{i*} \cdot \mathrm{diag}(B^{\mathrm{T}},1), S_{+1}) = (1 \pm \varepsilon) \sum_i \mathrm{dist}(A_{i*}, S).$$

Using the above relation, we give a coreset construction for the k-subspace approximation and k-median problems. These constructions are as in [SW18]. For any matrix M, let M_{+1} denote the matrix M with a new column of 0s appended at the end and let M_{-1} denote the matrix M with the last column deleted.

Theorem 4.6.1 (Coreset for Subspace Approximation)**.** *There exists a sampling-and-scaling matrix T that samples and scales $\widetilde{O}(k^3/\varepsilon^8)$ rows of the matrix Apx such that, with probability $\geq 3/5$, for any projection matrix P of rank k that projects onto a subspace S of dimension at most k, we have*

$$\begin{aligned} &\|((T \cdot Apx \cdot \mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1} - T \cdot Apx \cdot \mathrm{diag}(B^{\mathrm{T}},1)\|_{1,2} \\ &= (1 \pm O(\varepsilon))\|((Apx \cdot \mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1} - Apx \cdot \mathrm{diag}(B^{\mathrm{T}},1)\|_{1,2} \\ &= (1 \pm O(\varepsilon)) \sum_i \mathrm{dist}(A_i, S). \end{aligned}$$

This sampling matrix can be computed in time $O(n \cdot \mathrm{poly}(k/\varepsilon))$.

Proof. We first have

$$\begin{aligned} &\|((\mathrm{Apx} \cdot \mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1} - \mathrm{Apx} \cdot \mathrm{diag}(B^{\mathrm{T}},1)\|_{1,2} \\ &= \sum_i \|((\mathrm{Apx}_{i*} \cdot \mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1} - \mathrm{Apx}_{i*} \cdot \mathrm{diag}(B^{\mathrm{T}},1)\|_2 \\ &= \sum_i \sqrt{\|(I-P)BX_{i*}^{\mathrm{T}}\|_2^2 + v_i^2} \\ &= \sum_i \sqrt{\mathrm{dist}(BX_{i*}^{\mathrm{T}}, S)^2 + v_i^2} \\ &= (1 \pm \varepsilon) \sum_i \mathrm{dist}(A_{i*}, S). \end{aligned}$$

Let G be a Gaussian matrix with $\widetilde{O}(d/\varepsilon^2)$ columns. Then with probability $\geq 9/10$, for all $x \in \mathbb{R}^{d+1}$,

$$\|x^{\mathrm{T}}G\|_1 = (1 \pm \varepsilon)\|x\|_2.$$

See [SW18] for references. Thus, we have that with probability $\geq 9/10$, for all projection matrices P

of rank at most k, we have

$$\begin{aligned}
&\|((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}G-\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)G\|_{1,1}\\
&=(1\pm\varepsilon)\|((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}-\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)\|_{1,2}.
\end{aligned}$$

Note that for any P, the columns of the matrix $((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}G-\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)G$ lie in the column space of the matrix Apx. Let T be a $(1\pm\varepsilon)$ ℓ_1-subspace embedding constructed for the matrix Apx constructed using [CP15]. Therefore,

$$\begin{aligned}
&\|T\cdot((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}G-T\cdot\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)G\|_{1,1}\\
&=(1\pm\varepsilon)\|((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}G-\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)G\|_{1,1}.
\end{aligned}$$

Again, using the fact that $\|x^{\mathrm{T}}G\|_1=(1\pm\varepsilon)\|x\|_2$ for all $d+1$ dimensional vectors x, we obtain that

$$\begin{aligned}
&\|T\cdot((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}-T\cdot\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)\|_{1,2}\\
&=(1\pm\varepsilon)\|T\cdot((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}G-T\cdot\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)G\|_{1,1}\\
&=(1\pm O(\varepsilon))\|((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}G-\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)G\|_{1,1}\\
&=(1\pm O(\varepsilon))\|((\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1))_{-1}P)_{+1}-\mathrm{Apx}\cdot\mathrm{diag}(B^{\mathrm{T}},1)\|_{1,2}\\
&=(1\pm O(\varepsilon))\sum_i\mathrm{dist}(A_i,S).
\end{aligned}$$

The matrix T is computed by Lewis Weight Sampling. As the matrix Apx has dimensions $n\times\widetilde{O}(k^3/\varepsilon^6)$, we see from [CP15] that the matrix T can be computed in time $n\cdot\mathrm{poly}(k/\varepsilon)$. □

Theorem 4.6.2 (Coreset for k-median). *There exists a subset $T\subseteq[n]$ with $|T|=\widetilde{O}(k^4/\varepsilon^8)$ and weights w_i for $i\in T$ such that, with probability $\geq 3/5$, for any set C of size k,*

$$\sum_{i\in T}w_i\cdot\mathrm{dist}(Apx_{i*}\cdot diag(B^{\mathrm{T}},1),C_{+1})=(1\pm\varepsilon)\sum_{i\in[n]}\mathrm{dist}(A_{i*},C).$$

Recall that $C_{+1}=\{\,(c,0)\mid c\in C\,\}$.

Proof. Let S denote the rowspan of the matrix $\mathrm{diag}(B^{\mathrm{T}},1)$. We have $\dim(S)=\widetilde{O}(k^3/\varepsilon^6)$. Let $\hat{S}$ be the subspace S along with an orthogonal dimension. Thus, $\hat{S}$ is an $\widetilde{O}(k^3/\varepsilon^6)$ dimensional subspace of $\mathbb{R}^{d+1}$. Let $C=\{\,c_1,\ldots,c_k\,\}$ be an arbitrary set of k centers of $\mathbb{R}^{d+1}$. Now it is easy to see that we can find a set of k points $\hat{C}=\{\,\hat{c}_1,\ldots,\hat{c}_k\,\}\subseteq\hat{S}$ such that $\mathbb{P}_Sc_i=\mathbb{P}_S\hat{c}_i$ i.e., the projections of c_i and $\hat{c}_i$ onto the subspace S are the same, and also that $\mathrm{dist}(c_i,\mathbb{P}_S(c_i))=\mathrm{dist}(\hat{c}_i,\mathbb{P}_S(\hat{c}_i))$ and therefore, for any point $a\in S$, $\mathrm{dist}(a,C)=\mathrm{dist}(a,\hat{C})$.

Now if $T \subseteq [n]$ and the weights w_i for $i \in T$ are such that

$$\sum_{i \in T} w_i \cdot \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), \widetilde{C}) = (1 \pm \varepsilon) \sum_{i=1}^{n} \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), \widetilde{C})$$

for all k-center sets $\widetilde{C} \subseteq \hat{S}$, then for any k center set $C \subseteq \mathbb{R}^{d+1}$, we have

$$\begin{aligned}
\sum_{i \in T} w_i \cdot \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), C) &= \sum_{i \in T} w_i \cdot \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), \hat{C}) \\
&= (1 \pm \varepsilon) \sum_{i=1}^{n} \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), \hat{C}) \\
&= (1 \pm \varepsilon) \sum_{i=1}^{n} \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), C).
\end{aligned}$$

Thus, preserving the k-median distances with respect to the k center sets that lie in $\hat{S}$, preserves the k-median distances to all the center sets in $\mathbb{R}^{d+1}$. Using the coreset construction of [FL11] on the matrix Apx, we can obtain a subset $T \subseteq [n]$ of size $\widetilde{O}(k^4/\varepsilon^8)$ along with weights w_i such that for any k-center set $C \subseteq \mathbb{R}^{d+1}$, we have

$$\sum_{i \in T} w_i \cdot \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), C) = (1 \pm \varepsilon) \sum_{i=1}^{n} \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), C).$$

As Apx is an $n \times \operatorname{poly}(k/\varepsilon)$-sized matrix, the algorithm of [FL11] can be run in time $n \cdot \operatorname{poly}(k/\varepsilon)$. Thus, the above subset T and weights w_i for $i \in T$ can be found in time $n \operatorname{poly}(k/\varepsilon)$. Now, for any k-center set $C \subseteq \mathbb{R}^d$, we have that

$$\begin{aligned}
\sum_{i=1}^{n} \operatorname{dist}(A_{i*}, C) &= (1 \pm \varepsilon) \sum_{i=1}^{n} \sqrt{\operatorname{dist}(BX_{i*}^{\mathrm{T}}, C) + v_i^2} \\
&= (1 \pm \varepsilon) \sum_{i=1}^{n} \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), C_{+1}) \\
&= (1 \pm \varepsilon) \sum_{i \in T} w_i \cdot \operatorname{dist}(\mathrm{Apx}_{i*} \cdot \operatorname{diag}(B^{\mathrm{T}}, 1), C_{+1}).
\end{aligned}$$

Therefore, we obtain a coreset of size $\widetilde{O}(k^4/\varepsilon^8)$ in overall time $\widetilde{O}(\operatorname{nnz}(A)/\varepsilon^2 + (n+d) \operatorname{poly}(k/\varepsilon))$. $\square$

Theorem 4.6.3. *Given an $n \times d$ matrix A, $k \in \mathbb{Z}$, and an accuracy parameter $\varepsilon > 0$, Algorithm 4.4 returns a matrix B with $\widetilde{O}(k^3/\varepsilon^6)$ orthonormal columns and a matrix $\mathrm{Apx} = [X\ v]$ such that, with probability $\geq 9/10$, for any k dimensional shape S, $\sum_i \sqrt{\operatorname{dist}(BX_{i*}^{\mathrm{T}}, S)^2 + v_i^2} = (1 \pm \varepsilon) \sum_i \operatorname{dist}(A_i, S)$. The algorithm runs in time $O(\operatorname{nnz}(A)/\varepsilon^2 + (n+d) \operatorname{poly}(k/\varepsilon))$.*

Let B_i be the value of the matrix B after i iterations in Algorithm 4.3. The proof of the above theorem first shows that Algorithm 4.3 outputs a subspace B satisfying (4.1). This is done by showing that for at least a constant fraction of $j \in [10/\varepsilon+1]$, the terms $\|A(I-B_jB_j^{\mathrm{T}})\|_{1,2}$ and $\|A(I-B_{j+1}B_{j+1}^{\mathrm{T}})\|_{1,2}$ are close. This further means that the rows of the matrix $A(I-B_jB_j^{\mathrm{T}})$ cannot be projected onto any k dimensional subspace W to make $\|A(I-B_jB_j^{\mathrm{T}})(I-WW^{\mathrm{T}})\|_{1,2}$ substantially smaller than $\|A(I-B_jB_j^{\mathrm{T}})\|_{1,2}$. Thus, we can show that with constant probability, for i^* chosen randomly by Algorithm 4.3, the subspace $\mathrm{colspan}(B_{i^*})$ satisfies (4.1).

Then, the proof uses the fact that for every $i \in [n]$, the algorithm finds a matrix $\boldsymbol{S}_j$ that is a $\Theta(\varepsilon^2)$ subspace embedding for $[B\,A_{i*}^{\mathrm{T}}]$. This is shown to be true in [LBKW14]. Now, if $\boldsymbol{S}_j$ is a subspace embedding, it can be shown that the vector x_i and value v_i satisfy the conditions of Theorem 4.3.1, thus proving the above theorem.

4.7 Conclusions and Open Questions

In this work, we improve the construction of [SW18] to obtain dimensionality reduction procedure for the sum-of-distances objective that can be applied to an arbitrary matrix A in time $\widetilde{O}(\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k/\varepsilon))$.

The dimensionality reduction procedure of [SW18] and consequently ours appends an additional dimension to the points that turns out to be important to be able to approximate the sum-of-distances to a k-dimensional object. For clustering problems, Huang and Vishnoi [HV20] obtain coresets without the need to append an additional coordinate. An interesting question is if we can obtain such a result for the general sum-of-distances objective.

Chapter 5

Lower Bounds on Adaptive Sensing for Matrix Recovery

5.1 Introduction

Sparse recovery, also known as compressed sensing, is the study of under-determined systems of equations subject to a sparsity constraint. Suppose we know that an unknown vector $x \in \mathbb{R}^n$ has at most $r \ll n$ non-zero entries. We would like to use a measurement matrix $M \in \mathbb{R}^{k\times n}$ to recover the vector x given measurements $y = Mx$. The number k of measurements is an important parameter we would like to optimize as it models the equipment cost in physical settings and running times in computational settings. Ideally one would like for the number of measurements k to scale proportionally with the sparsity of the unknown vector x.

A more robust way of modeling sparse recovery is as "stable sparse recovery". Here we want a distribution $\mathcal{M}$ over $k \times n$ matrices such that for any $x \in \mathbb{R}^n$, with probability $\geq 1 - \delta$ over $\boldsymbol{M} \sim \mathcal{M}$, we can construct a vector $\hat{x}$ from $\boldsymbol{M}x$ such that

$$\|x - \hat{x}\|_p \leq (1+\varepsilon) \min_{r\text{-sparse } x'} \|x - x'\|_p.$$

This formulation does not require the underlying vector x to be exactly r-sparse and instead asks to recover the top r coordinates of x.

For $p = 2$, it is known that $k = \Theta(\varepsilon^{-1} r \log(n/r))$ measurements re both necessary and sufficient [CRT06, GLPS12, PW11, CD13]. See [PW12] for upper and lower bounds for other values of p.

In the same vein, the problem of low rank matrix recovery has been studied. See [CP11, ZJD15] and references therein for earlier work and numerous applications. In this problem, the aim is to recover a *low rank* matrix A using linear measurements. Here we want a distribution $\mathcal{M}$ over linear operators $M : \mathbb{R}^{n\times n} \to \mathbb{R}^k$, such that for any $n \times n$ *low* rank matrix A, with probability $\geq 1 - \delta$ over $\boldsymbol{M} \sim \mathcal{M}$, we can construct a matrix $\hat{A}$ from $\boldsymbol{M}(A)$ such that the reconstruction error $\|A - \hat{A}\|_\mathrm{F}^2$

is small with a large probability. Note that a linear operator $M : \mathbb{R}^{n\times n} \to \mathbb{R}^k$ can be equivalently represented as a matrix $M \in \mathbb{R}^{k\times n^2}$ such that $M \cdot \text{vec}(A) = M(A)$ where $\text{vec}(A)$ denotes the appropriate flattening of the matrix A into a vector. We call the rows of M *linear measurements*. Without loss of generality, we can assume that the rows of the matrix M are orthonormal, as the responses for non-orthonormal queries can be obtained via a simple change of basis.

In addition, to model the measurement error that occurs in practice, it is a standard assumption that when querying with M, we receive $M \cdot \text{vec}(A) + \boldsymbol{g}$, where $\boldsymbol{g}$ is a k-dimensional vector with independent Gaussian random variables of mean 0 and variance σ^2, and we hope to reconstruct A with small error from $M \cdot \text{vec}(A) + \boldsymbol{g}$. Clearly, when A has rank r, we need to perform $\Omega(nr)$ linear measurements, as the matrix A has $\Theta(nr)$ independent parameters. Hence, the aim is to perform not many more linear measurements than nr while being able to obtain an estimate $\hat{A}$ for A with low estimation error.

Given a rank-r matrix A, [CP11] show that if the $k\times n^2$ matrix $\boldsymbol{M}$ is constructed with independent standard Gaussian entries, then with probability $\geq 1 - \exp(-cn)$, an estimate $\hat{A}$ can be constructed from $\boldsymbol{M} \cdot \text{vec}(A) + \boldsymbol{g}$ such that $\|A - \hat{A}\|_{\text{F}}^2 \leq O(nr\sigma^2/k)$. They use the restricted isometry property (RIP) of the Gaussian matrix $\boldsymbol{M}$ to obtain algorithms that give an estimate $\hat{A}$. The error bound is intuitive since the reconstruction error increases with increasing noise and proportionally goes down when the number of measurements k increases.

While we formulated the sparse recovery and matrix recovery problems in a non-adaptive way, there have been works which study adaptive algorithms for sparse recovery. Here we can produce matrices $\boldsymbol{M}^{(i)}$ adaptively based on the responses received $\boldsymbol{M}^{(1)}x, \boldsymbol{M}^{(2)}x, \ldots, \boldsymbol{M}^{(i-1)}x$ and the hope is that allowing for adaptive algorithms with a small number of adaptive rounds, we obtain algorithms that overall perform fewer linear measurements than non-adaptive algorithms with the same reconstruction error. It is additionally assumed that the noise across different rounds is independent. For sparse recovery, in the case of $p = 2$, it is known that over $O(\log\log n)$ rounds adaptivity, a total of $O(\varepsilon^{-1} r \log\log(n \log(n/r)))$ linear measurements suffices [IPW11, NSWZ18], improving over the requirement of $\Theta(\varepsilon^{-1} r \log(n/r))$ linear measurements for non-adaptive algorithms.

While there has been a lot of interest in adaptive sparse recovery, both from the algorithms and the lower bounds perspective (such as [PW12, KP19]), the adaptive matrix recovery problem surprisingly does not seem to have any known lower bounds. In adaptive matrix recovery, similar to adaptive sparse recovery, one is allowed to query a measurement matrix $M^{(i)}$ in round i based on the responses received in the previous rounds, and again the hope is that with more rounds of adaptivity, the total number of linear measurements that is to be performed decreases. There is some work that studies adaptive matrix recovery with 2 rounds of adaptivity (see [ZKXL19] and references therein) but the full landscape of adaptive algorithms for matrix recovery seems unexplored.

We address this from the lower bounds side in this work. We show lower bounds on adaptive algorithms that recover a rank-r matrix of the form $(\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\text{T}}$, where the coordinates of $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are sampled independently from the standard Gaussian distribution. Without loss of generality[1], we assume that the measurement matrix $M^{(i)}$ in each round i has orthonormal rows. We also assume that for $i \neq j$, the measurement matrices $M^{(i)}(M^{(j)})^{\mathrm{T}} = 0$, i.e., measurements across adaptive rounds are orthonormal, since non-orthonormal measurements can be reconstructed from orthonormal measurement matrices by a change of basis.

We now give an alternate way of looking at the adaptive sparse recovery problem: Fix a set of vectors $u_1, \ldots, u_r$ and $v_1, \ldots, v_r$ and let the underlying matrix to be reconstructed is $(\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}$ for a large enough constant α. In the first round, we query a matrix $\boldsymbol{M}^{(1)} \in \mathbb{R}^{k \times n^2}$ drawn from an appropriate distribution and receive the response vector

$$\boldsymbol{r}^{(1)} = \boldsymbol{M}^{(1)} \operatorname{vec}((\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}) + \boldsymbol{g}^{(1)}$$

where $\boldsymbol{g}^{(1)}$ is a vector of independent Gaussian random variables with mean 0 and variance σ^2. In round 2, as a function of $\boldsymbol{M}^{(1)}$ and $\boldsymbol{r}^{(1)}$ and our randomness, we query a random matrix $\boldsymbol{M}^{(2)}[\boldsymbol{r}^{(1)}] \in \mathbb{R}^{k \times n^2}$ and receive a response vector $\boldsymbol{r}^{(2)} = (\boldsymbol{M}^{(2)}[\boldsymbol{r}^{(1)}]) \operatorname{vec}((\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}) + \boldsymbol{g}^{(2)}$ where $\boldsymbol{g}^{(2)}$ is again a vector with independent Gaussian entries and independent of $\boldsymbol{g}^{(1)}$, and so on. Crucially, using the rotational invariance of Gaussian random vectors, if $\boldsymbol{G}$ is an $n \times n$ matrix with independent Gaussian random variables with mean 0 and variance σ^2, the response $\boldsymbol{r}^{(1)}$ has the same distribution as $(\boldsymbol{M}^{(1)}) \cdot \operatorname{vec}((\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}} + \boldsymbol{G})$ and as $\boldsymbol{M}^{(2)}[\boldsymbol{r}^{(1)}]$ is chosen to be orthonormal to $\boldsymbol{M}^{(1)}$, the distribution of $\boldsymbol{r}^{(2)}$ conditioned on $\boldsymbol{r}^{(1)}$ is the same as that of $(\boldsymbol{M}^{(2)}[\boldsymbol{r}^{(1)}]) \cdot (\operatorname{vec}((\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}} + \boldsymbol{G}))$.

Thus, any adaptive matrix recovery algorithm can be seen as performing *non-noisy* adaptive measurements on the matrix $(\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}} + \boldsymbol{G}$ where the Gaussian matrix $\boldsymbol{G}$ is sampled independently of the measurement algorithm. From the responses the algorithm receives, it then tries to reconstruct the matrix $(\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}$. This way of looking at adaptive sparse recovery immediately yields an adaptive algorithm: when the smallest singular value of $(\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}$ is a constant times larger than $\|\boldsymbol{G}\|_2$, then the power method with a block size of r outputs an approximation of $(\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}$ in $O(\log n)$ rounds. Note that r matrix-vector products can be implemented using nr linear measurements. More generally, Gu [Gu15] showed that using power iteration with a block size of k/n (for $k \geq nr$), we can obtain an approximation in $O(\log(n^2/k))$ adaptive rounds. Thus, the already existing algorithms exhibit a # of measurements vs # of rounds trade-off.

From results in random matrix theory, we have that $\|\boldsymbol{G}\|_2 \approx \sigma\sqrt{n}$ with high probability. And as we are interested in reconstruction when the vectors $u_1, \ldots, u_r$ and $v_1, \ldots, v_r$ follow the Gaussian distribution we also have that $\|u_i\|_2 \approx \|v_i\|_2 \approx \sqrt{n}$ simultaneously with large probability. Thus, to make the extraction of $(\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}$ possible, we need to assume that $\alpha \gtrsim \sigma$. Hence, in this

[1]In general, in the sparse recovery and matrix recovery models, the algorithms may make the same query and obtain responses with independent noise added which can be used to say obtain more accurate result using the median/mean estimator. But all the algorithms we are aware of for sparse recovery and matrix recovery do not explore independence of noise across queries in this way.

work we assume that $\sigma = 1$ and that α is a large enough constant, and we study lower bounds on adaptive matrix recovery algorithms.

If the vectors $u_1, \dots, u_r$ and $v_1, \dots, v_r$ are sampled from the standard n dimensional Gaussian distribution and $r \le n/2$, we also have that with high probability $\|(\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \approx \alpha^2 nr$. Now the algorithms of [CP11], which use a uniform random Gaussian matrix $\boldsymbol{M}$ with m rows as the measurement matrix, for $(\alpha/\sqrt{n}) \sum_{i=1} u_i v_i^{\mathrm{T}}$ reconstruct a matrix $\hat{A}$ such that $\|\hat{A} - (\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \le C\frac{nr\sigma_m^2}{m}$ where σ_m is the measurement error. As we assumed above that $\sigma = 1$ when measuring with a matrix with orthonormal rows, we assume that the measurement error σ_m when measuring with a Gaussian matrix is n, as each row of $\boldsymbol{M}$ has n^2 independent Gaussian coordinates and therefore has a norm $\approx n$. Thus, using reconstruction algorithms from [CP11], we obtain a matrix $\hat{A}$ satisfying $\|\hat{A} - (\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \le C\frac{n^3 r}{m}$. Now, to make $\|\hat{A} - (\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \le (1/10)\|(\alpha/\sqrt{n}) \sum_{i=1}^{r} u_i v_i^{\mathrm{T}}\|_{\mathrm{F}}^2$, we need to set $m = \Theta(n^2/\alpha^2)$. Hence, in the parameter regimes we study, the algorithms of [CP11] need to perform $\Omega(n^2)$ non-adaptive queries, which essentially means that they have to read a constant fraction of the n^2 entries in the matrix, whereas the power method performs $O(nr)$ linear measurements in each round over $O(\log n)$ adaptive rounds to output an approximation $\hat{A}$. The question we ask is:

"Is the power method optimal? Are there algorithms that have $o(\log n)$ adaptive rounds and use $n^{2-\beta}$ measurements in each round?"

We answer this question by showing that any algorithm that has $o(\log n/\log\log n)$ adaptive rounds must use $\ge n^{2-\beta}$ linear measurements, for any constant $\beta > 0$, in each round. The lower bound shows that power method is essentially optimal if we want to use $n^{2-\beta}$ linear measurements in each round and that any algorithm with $o(\log n/\log\log n)$ adaptive rounds essentially reads the whole matrix.

We further obtain a rounds vs measurements trade-off for many numerical linear algebra problems in the sensing model. In this model, there is an $n \times n$ matrix A with which we can interact only using general linear measurements and we want to solve problems such as spectral norm low rank approximation of A, Frobenius norm low rank approximation of A, etc. In general, numerical linear algebra algorithms assume that they either have access to the entire matrix or that the matrix is accessible in the matrix-vector product model where one can query a vector v and obtain the result $A \cdot v$. Recently, the vector-matrix-vector product model has received significant attention as well. Linear measurements are more powerful than both the matrix-vector product model and the vector-matrix-vector product model. Any matrix vector product $A \cdot v$ can be computed using n linear measurements of A and any vector-matrix-vector product $u^{\mathrm{T}}Av$ can be computed using a single linear measurement of A. Thus, the model of general linear measurements may lead to faster algorithms.

For certain problems in numerical linear algebra, general linear measurements are significantly more powerful than the matrix-vector product model. Indeed, for computing the trace of an $n \times n$ matrix A, one can do this exactly with a single deterministic general linear measurement, just by adding up the diagonal entries of A. However, in the matrix-vector product model, it is known that $\Omega(n)$ matrix-vector products are needed to compute the trace exactly [SWYZ21b], even if one uses randomization. A number of problems were studied in the vector-matrix-vector product model in [RWZ20], and in [WWZ14] it was shown that to approximate the trace of A up to a $(1+\varepsilon)$-factor with probability $1-\delta$, one needs $\Omega(\varepsilon^{-2}\log(1/\delta))$ queries. This contrasts sharply with the single deterministic general linear measurement for computing the trace exactly. Thus, there are good reasons to conjecture that general linear measurements may lead to algorithms requiring fewer rounds of adaptivity compared to algorithms in the matrix-vector product query model. Surprisingly, our lower bounds show that for many numerical linear algebra problems, linear measurements do not give much advantage over matrix-vector products.

5.1.1 Our Results

We assume that there is an unknown rank-r matrix $A \in \mathbb{R}^{n\times n}$ to be recovered. Given any linear measurement $q \in \mathbb{R}^{n^2}$, we receive a response $\langle q, \mathrm{vec}(A)\rangle + \boldsymbol{g}$, where $\boldsymbol{g} \sim N(0, \|q\|_2^2)$. We further assume that the noise for two different measurements is independent. Without loss of generality, we also assume throughout that all the queries an algorithm makes across all rounds form a matrix with orthonormal rows. Our main result for adaptive matrix sensing is as follows:

Theorem 5.1.1. *There exists a constant c such that any randomized algorithm which makes $k \geq nr$ noisy linear measurements of an arbitrary rank-r matrix A with $\|A\|_F^2 = \Theta(nr)$ in each of t rounds, and outputs an estimate $\hat{A}$ satisfying $\|A-\hat{A}\|_F^2 \leq c\|A\|_F^2$ with probability $\geq 9/10$ over the randomness of the algorithm and the Gaussian noise, requires $t = \Omega(\log(n^2/k)/(\log\log n))$.*

Dependence on noise In our results, we assumed that given a linear measurement $q \in \mathbb{R}^{n^2}$, the response is distributed as $N(\langle q, \mathrm{vec}(A)\rangle, \|q\|_2^2)$. Our lower bounds also hold when the response is distributed as $N(\langle q, \mathrm{vec}(A)\rangle, \sigma^2\|q\|_2^2)$ for any parameter σ such that $c' \leq \sigma \leq 1$, where c' is a constant. This can be seen by simply scaling the matrix A in the theorem above and adjusting the constants while proving the theorem.

Tensor Recovery The problem of tensor recovery with linear measurements has also been studied (see [RSS17, GLM+21] and references therein) where given a low rank tensor, the task is to recover an approximation to the tensor with few linear measurements. Our techniques can potentially be used to obtain lower bounds on adaptive algorithms for tensor recovery. Our main tool, Lemma 5.3.1, readily extends to tensors of higher orders by using the corresponding tail bounds from [Ver20].

Numerical Linear Algebra We also derive lower bounds for many numerical linear algebra problems in the linear measurements model. Table 5.1 shows our lower bounds on the number of adaptive

Application	Failure Probability	Lower Bound
2-approximate spectral LRA	0.1	$c \log(n^2/k)/\log\log n$
2-approximate spectral LRA	$1/\text{poly}(n)$	$c \log(n^2/k)$
2-approximate Schatten p LRA	0.1	$\frac{c\log(n^2/k)}{(1/p)\log(n)+\log\log n}$
2-approximate Ky-Fan p LRA	0.1	$\frac{c\log(n^2/k)}{\log(p)+\log\log n}$
$1 + 1/n$-approximate Frobenius LRA	0.1	$c \log(n^2/k)/\log\log n$
0.1-approximate i-th singular vectors	0.1	$c \log(n^2/k)/\log\log n$
2-approximate spectral reduced rank regression	0.1	$c \log(n^2/k)/\log\log n$

Table 5.1: Number of rounds lower bound for algorithms using k general linear measurements in each round. Our bound for 2-approximate spectral LRA algorithm with constant failure probability is optimal up to a $\log\log n$ factor, and our 2-approximate spectral low rank approximation (LRA) lower bound for algorithms with failure probability $1/\text{poly}(n)$ is optimal up to a constant factor.

rounds required for any randomized algorithm using k general linear measurements in each round. See Section 5.5 for precise statements and proofs.

5.1.2 Notation

For vectors $u, v \in \mathbb{R}^n$, $u \otimes v \in \mathbb{R}^{n^2}$ denotes the tensor product of u and v. For an arbitrary matrix $M \in \mathbb{R}^{m\times n}$, the vector $\text{vec}(M) \in \mathbb{R}^{mn}$ denotes a flattening of the matrix M with the convention that $\text{vec}(uv^{\mathrm{T}}) = u \otimes v$. We use $\boldsymbol{g}_k$ to denote a multivariate Gaussian in k dimensions where each coordinate is independently sampled from $N(0, 1)$. We also use $\boldsymbol{G}^{(j)}$ to denote a collection of j independent multivariate Gaussian random variables of appropriate dimensions.

5.1.3 Our Techniques

Using the rotational invariance of the Gaussian distribution, we argued that any adaptive randomized low rank matrix recovery algorithm with access to a hidden matrix $(\alpha/\sqrt{n})\sum_{i=1}^{r} u_i v_i^{\mathrm{T}}$ using noisy linear measurements can be seen as a randomized algorithm that has access to a random matrix $(\alpha/\sqrt{n})\sum_{i=1}^{r} u_i v_i^{\mathrm{T}} + \boldsymbol{G}$ using *perfect* linear measurements where each coordinate of $\boldsymbol{G}$ is independently sampled from a Gaussian distribution.

Let $\mathcal{A}$ be any algorithm that satisfies the matrix recovery guarantees with, say, a success probability $\geq 9/10$. Let $\mathcal{A}(X, \boldsymbol{\sigma}, \boldsymbol{\gamma})$ be the output of the randomized algorithm $\mathcal{A}$, where the hidden matrix is X, the random seed of $\mathcal{A}$ is denoted by $\boldsymbol{\sigma}$, and $\boldsymbol{\gamma}$ denotes the measurement randomness. We have that if X has rank r and satisfies $\|X\|_{\mathrm{F}}^2 = \Theta(nr)$, then

$$\mathbf{Pr}_{\sigma,\gamma}[\mathcal{A}(X, \boldsymbol{\sigma}, \boldsymbol{\gamma}) \text{ is correct}] \geq 9/10.$$

We say $\mathcal{A}(X, \boldsymbol{\sigma}, \boldsymbol{\gamma})$ is correct when the output $\hat{X}$ satisfies $\|\hat{X} - X\|_{\mathrm{F}}^2 \leq (1/100)\|X\|_{\mathrm{F}}^2$. By the above reduction, from $\mathcal{A}$ we have a randomized algorithm $\mathcal{A}'$ that runs on the random matrix $X + \boldsymbol{G}$ with access to exact linear measurements and outputs a correct reconstruction $\hat{X}$ with probability $\geq 9/10$ if X has rank r and $\|X\|_{\mathrm{F}}^2 = \Theta(nr)$. Thus, for all such X,

$$\mathbf{Pr}_{G,\sigma}[\mathcal{A}'(X + \boldsymbol{G}, \boldsymbol{\sigma}) \text{ is correct}] \geq 9/10.$$

Here $\boldsymbol{\sigma}$ denotes the randomness used by the algorithm $\mathcal{A}'$. Now, if $\boldsymbol{X}$ is a random matrix constructed as $(\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ with $\boldsymbol{u}_i, \boldsymbol{v}_i$ being random vectors with independent Gaussian entries of mean 0 and variance 1, then with probability $\geq 99/100$, $\|\boldsymbol{X}\|_{\mathrm{F}}^2 = \Theta(nr)$. Thus,

$$\mathbf{Pr}_{X,G,\sigma}[\mathcal{A}'(\boldsymbol{X} + \boldsymbol{G}, \boldsymbol{\sigma}) \text{ is correct}] \geq 8/10.$$

Hence, there exists some fixed σ such that

$$\mathbf{Pr}_{X,G}[\mathcal{A}'(\boldsymbol{X} + \boldsymbol{G}, \sigma) \text{ is correct}] \geq 8/10.$$

Thus, the existence of a randomized algorithm that solves low rank matrix recovery as in Theorem 5.1.1 implies the existence of a deterministic algorithm which given access to perfect linear measurements of random matrix $(\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} + \boldsymbol{G}$ outputs a reconstruction of $(\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ with probability $\geq 8/10$. This is essentially a reduction from randomized algorithms to deterministic algorithms using Yao's lemma. From here on, we prove lower bounds on such deterministic algorithms and conclude the lower bounds in Theorem 5.1.1. For simplicity, we explain the proof of our lower bound for $r = 1$ here and extend to general r later. We consider the random matrix $\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ and show how the lower bound proof proceeds.

Note that the matrix $\boldsymbol{A} \coloneqq \boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} \in \mathbb{R}^{n \times n}$ can be flattened to the vector $\boldsymbol{a} = \boldsymbol{g} + (\alpha/\sqrt{n})(\boldsymbol{u} \otimes \boldsymbol{v}) \in \mathbb{R}^{n^2}$. Also, a general linear measurement, which we call a query $Q \in \mathbb{R}^{n \times n}$, can be vectorized to $q = \mathrm{vec}(Q) \in \mathbb{R}^{n^2}$ with $Q(\boldsymbol{A}) = \langle q, \boldsymbol{a} \rangle$. Fix any deterministic algorithm. In the first round, the algorithm starts with a fixed matrix $Q^{(1)} \in \mathbb{R}^{k \times n^2}$ that corresponds to the k queries and receives the response $\boldsymbol{r}^{(1)} \coloneqq Q^{(1)}\boldsymbol{a}$. Then, as a function of the response $\boldsymbol{r}^{(1)}$ in the first round, the algorithm picks a matrix $Q^{(2)}[\boldsymbol{r}^{(1)}]$ in the second round and receives the response $\boldsymbol{r}^{(2)} \coloneqq Q^{(2)}[\boldsymbol{r}^{(1)}] \cdot \boldsymbol{a}$. Similarly, in the i-th round, the deterministic algorithm picks a matrix $Q^{(i)}[\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}] \in \mathbb{R}^{k \times n^2}$ as a function of $\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}$ and receives the response $\boldsymbol{r}^{(i)} \coloneqq Q^{(i)}[\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}] \cdot \boldsymbol{a}$. Note that we assumed that the query matrices $Q^{(i)}$ chosen by the algorithm have orthonormal rows and also that $Q^{(i)}(Q^{(j)})^{\mathrm{T}} = 0$, i.e., the queries across rounds are also orthonormal.

For a fixed $u, v \in \mathbb{R}^n$, we see that the response $\boldsymbol{r}^{(1)} = Q^{(1)}\boldsymbol{g} + (\alpha/\sqrt{n})Q^{(1)}(u \otimes v)$. As the matrix $Q^{(1)}$ has orthonormal rows, the random variable $Q^{(1)}\boldsymbol{g} \equiv \boldsymbol{g}_k$, where $\boldsymbol{g}_k \sim N(0, I_k)$ is drawn from a mean-zero normal distribution with identity covariance. Thus, for fixed u, v, the distribution of the first round responses to the algorithm is $N((\alpha/\sqrt{n})Q^{(1)}(u \otimes v), I_k)$. Now the key observation is that $\|(\alpha/\sqrt{n})Q^{(1)}(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 = \Theta(\alpha^2 k/n)$ with high probability over the inputs $(\boldsymbol{u}, \boldsymbol{v})$. This uses a recent concentration result for random tensors due to [Ver20], and critically uses the fact that $Q^{(1)}$ has operator norm 1 and Frobenius norm $\sqrt{k}$. This means that for a large fraction of (u, v) pairs, the distribution of the responses seen by the algorithm in the first round is close to $N(0, I_k)$, and therefore just looking at the response $\boldsymbol{r}^{(1)}$, the algorithm cannot have a lot of "information" about which (u, v) pair is involved in the matrix that is unknown to the algorithm. So the query chosen in the second round $Q^{(2)}[\boldsymbol{r}^{(1)}]$ cannot have a "large" value of $Q^{(2)}[\boldsymbol{r}^{(1)}](u \otimes v)$ for most inputs (u, v), with a high probability over the Gaussian component of the matrix.

Suppose the value of $Q^{(2)}[\boldsymbol{r}^{(1)}](u \otimes v)$ is small. Again, $\boldsymbol{r}^{(2)} = Q^{(2)}[\boldsymbol{r}^{(1)}]\boldsymbol{g} + (\alpha/\sqrt{n})Q^{(2)}[\boldsymbol{r}^{(1)}](u \otimes v)$. Crucially, as the queries in round 2 are orthogonal to the queries in round 1, we have by the rotational invariance of the Gaussian distribution that $Q^{(2)}[\boldsymbol{r}^{(1)}]\boldsymbol{g}$ is independent of $Q^{(1)}\boldsymbol{g}$, and that $Q^{(2)}[\boldsymbol{r}^{(1)}]\boldsymbol{g}$ is distributed as $N(0, I_k)$. So, for a fixed u, v, conditioned on the first round response $\boldsymbol{r}^{(1)}$, the distribution of the second round response $\boldsymbol{r}^{(2)}$ is given by $N((\alpha/\sqrt{n})Q^{(2)}[\boldsymbol{r}^{(1)}](u \otimes v), I_k) \approx N(0, I_k)$ using the assumption that $Q^{(2)}[\boldsymbol{r}^{(1)}](u \otimes v)$ is small. We again have that for a large fraction of pairs (u, v) for which $Q^{(2)}[\boldsymbol{r}^{(1)}](u \otimes v)$ is small, the distribution of the second round response is also close to $N(0, I_k)$. Therefore, the algorithm again does not gain a lot of information about which (u, v) pair is involved in the matrix, and the third round query $Q^{(3)}[\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(2)}]$ cannot have a large value of $\|Q^{(3)}[\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(2)}](u \otimes v)\|_2$. The proof proceeds similarly for further rounds and shows the necessity of a large number of adaptive rounds.

To formalize the above intuitive idea, we use Bayes risk lower bounds [CGZ16]. We show that with a large probability over the input matrix, the squared projection of $\boldsymbol{u} \otimes \boldsymbol{v}$ onto the query space of the algorithm is upper bounded and we use an iterative argument to show that an upper bound on the information up until round i can in turn be used to upper bound the information up until round $i + 1$. Bayes risk bounds are very general and let us obtain upper bounds on the information learned by a deterministic learner. Concretely, let Θ be a parameter space and $\mathscr{P} = \{ P_\theta : \theta \in \Theta \}$ be a set of distributions, one for each $\theta \in \Theta$. Let w be a distribution over Θ. We sample $\boldsymbol{\theta} \sim w$ and then $\boldsymbol{x} \sim P_{\boldsymbol{\theta}}$ and provide the learner with $\boldsymbol{x}$. Given an action space $\mathscr{A}$, the learner uses a deterministic decision rule $\mathfrak{d} : \mathcal{X} \to \mathscr{A}$ to minimize a 0-1 loss function $L : \Theta \times \mathscr{A} \to \{0, 1\}$ in expectation, i.e., $\mathbf{E}_{\boldsymbol{\theta} \sim w}[\mathbf{E}_{\boldsymbol{x} \sim P_{\boldsymbol{\theta}}} L(\boldsymbol{\theta}, \mathfrak{d}(\boldsymbol{x}))]$. Let $R_{\text{Bayes}}(L, \Theta, w) = \inf_{\mathfrak{d}} E_{\boldsymbol{\theta} \sim w}[E_{\boldsymbol{x} \sim P_{\boldsymbol{\theta}}} L(\boldsymbol{\theta}, \mathfrak{d}(\boldsymbol{x}))]$ be the loss achievable by the best deterministic decision rule $\mathfrak{d}$. Bayes risk lower bounds let us obtain lower bounds on R_{Bayes}.

In our setting after round 1, we have $\Theta = \{ (u, v) : u, v \in \mathbb{R}^n \}$, w is the joint distribution of two independent standard Gaussian random variables in $\mathbb{R}^n$ and for each $(u, v) \in \Theta$ we let P_{uv} be the distribution of $\boldsymbol{r}^{(1)} = \boldsymbol{g}_k + (\alpha/\sqrt{n})Q^{(1)}(u \otimes v)$, an action space $\mathscr{A}$ of all $k \times n^2$ matrices with orthonormal rows (corresponding to the queries in the next round), and define a 0-1 loss function

$$L((u, v), Q) = \begin{cases} 0 & \text{if } \|Q(u \otimes v)\|_2^2 \geq T \\ 1 & \text{if } \|Q(u \otimes v)\|_2^2 < T \end{cases}$$

for an appropriate threshold parameter T. By setting T appropriately as a function of t, we obtain a Bayes risk lower bound of $R_{\text{Bayes}} \ge 1-1/(100t^2)$. Thus, we obtain that for any deterministic decision rule $\mathfrak{d}$, with probability $\ge 1-1/10t$ over $(\boldsymbol{u}, \boldsymbol{v}) \sim w$, we have

$$\mathbf{E}_{\boldsymbol{r}^{(1)} \sim P_{\boldsymbol{uv}}}[L((\boldsymbol{u}, \boldsymbol{v}), \mathfrak{d}(\boldsymbol{r}^{(1)}))] \ge 1 - 1/10t$$

and in particular, we have that with probability $\ge 1-1/10t$ over $(\boldsymbol{u}, \boldsymbol{v}) \sim w$,

$$\mathbf{Pr}_{\boldsymbol{r}^{(1)} \sim P_{\boldsymbol{uv}}}[\|Q^{(2)}[\boldsymbol{r}^{(1)}](\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 < T] \ge 1 - 1/10t. \tag{5.1}$$

The above statement essentially says that with probability $\ge 1-1/10t$ over the inputs $(\boldsymbol{u}, \boldsymbol{v})$, the second query $Q^{(2)}[\boldsymbol{r}^{(1)}]$ chosen by the deterministic algorithm has the property that $\|Q^{(2)}[\boldsymbol{r}^{(1)}](\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 < T$ with probability $\ge 1-1/10t$ over the Gaussian $\boldsymbol{G}$. In the second round, we restrict our analysis of the algorithm to only those (u, v) which satisfy (5.1). We again define a distribution w' over the inputs and for each such (u, v) we define a distribution P_{uv} over the round 1 and round 2 responses received by the algorithm. We define a new loss function with parameter $T' = \Delta \cdot T$ for a multiplicative factor Δ and again obtain a statement similar to (5.1) for a large fraction of inputs (u, v) and repeat a similar argument for t rounds and show that there is an $\Omega(1)$ fraction of the inputs for which the squared norm of the projection of $u \otimes v$ onto the query space after t rounds is bounded by $T(t)$ with high probability over the Gaussian part of the input. This gives the result in Theorem 5.3.4. Note that $\|\boldsymbol{u} \otimes \boldsymbol{v}\|_2^2 = \Omega(n^2)$ with high probability and for any fixed matrix Q with k orthonormal rows, $\mathbf{E}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2] = k$ which corresponds to the amount of "information" the algorithm starts with. As we show that the "information" in each round grows by some multiplicative factor Δ, a number $\Omega(\log_\Delta(n^2/k))$ of rounds is required to obtain an "information" of $\Theta(n^2)$, which is how we obtain our lower bounds. Here information is measured as the squared projection of $\boldsymbol{u} \otimes \boldsymbol{v}$ onto the query space of the algorithm.

We also extend our results to identifying a rank r spike (sum of r random outer products) corrupted by Gaussian noise. Specifically, we consider the random matrix $\boldsymbol{M} = \boldsymbol{G} + (\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ where all the coordinates of $\boldsymbol{G}, \boldsymbol{u}_i, \boldsymbol{v}_i$ are independently sampled from $N(0, 1)$. We show that if there is an algorithm that uses k iterations in each round and identifies the spike with probability $\ge 9/10$, then it must run for $\Omega(\log(n^2/k)/\log\log n)$ rounds by appealing to the lower bound we described above for the case of $r = 1$. We show that if there is an algorithm for $r > 1$ that requires $t < O(\log(n^2/k)/\log\log n)$ adaptive rounds, then it can be used to solve the rank 1 spike estimation problem in t rounds as well which contradicts the lower bound.

We then provide lower bounds on approximate algorithms for a host of problems such as spectral norm low rank approximation (LRA), Schatten norm LRA, Ky-Fan norm LRA and reduced rank regression, by showing that algorithms to solve these problems can be used to estimate the spike $\boldsymbol{uv}^{\mathrm{T}}$ in the random matrix $\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{uv}^{\mathrm{T}}$, and then use the aforementioned lower bounds on algorithms that can estimate the spike. Although our hard distribution is supported on non-symmetric matrices, we are still able to obtain lower bounds for algorithms for spectral norm LRA (rank $r \geq 2$) for symmetric matrices as well by considering a suitably defined symmetric random matrix using our hard distribution. Let $\boldsymbol{A} \coloneqq \boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ be the hard distribution in the case of rank $r = 1$. We symmetrize the matrix by considering, $\boldsymbol{A}_{\mathrm{sym}} = \begin{bmatrix} 0 & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}} & 0 \end{bmatrix}$. This symmetrization, as opposed to $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ or $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, has the advantage that a linear measurement of $\boldsymbol{A}_{\mathrm{sym}}$ can be obtained from an appropriately transformed linear measurement of $\boldsymbol{A}$, thereby letting us obtain lower bounds for symmetric instances as well in the linear measurements model. However, we cannot obtain lower bounds for rank 1 spectral norm LRA for symmetric matrices using this symmetrization as the top two singular values of $\boldsymbol{A}_{\mathrm{sym}}$ are equal and hence even a zero matrix is a perfect rank 1 spectral norm LRA for $\boldsymbol{A}_{\mathrm{sym}}$ which does not give any information about the plant $\boldsymbol{u} \otimes \boldsymbol{v}$.

5.1.4 Related Work

As discussed, low rank matrix recovery has been extensively studied (see [CP11] and references therein for earlier work). Relatedly, [HFB15] study the Robust PCA problem where the aim is to estimate the sum of a low rank matrix and a sparse matrix from noisy vector products with the hidden matrix. [TV23] study the Robust PCA problem when given access to linear measurements with the hidden matrix.

For non-adaptive algorithms for low rank matrix recovery with Gaussian errors, [CP11] show that their selectors based on the restricted isometry property of measurement matrices are optimal up to constant factors in the minimax error sense when the noise follows a Gaussian distribution. Our Theorem 5.1.1 extends their lower bounds and shows that if there is any randomized measurement matrix[2] $\boldsymbol{M}$ with k rows coupled with a recovery algorithm that outputs a reconstruction for any rank r matrix with $\|A\|_{\mathrm{F}}^2 = \Theta(nr)$, then it must have $k = \Omega(n^{2-o(1)})$. We again note that we give lower bounds even for algorithms with multiple adaptive rounds.

Our technique to show lower bounds is to plant a low rank matrix $(\alpha/\sqrt{n})\sum_{i=1}^{r} u_i v_i^{\mathrm{T}}$ in an $n \times n$ Gaussian matrix so that any "orthonormal" access to the plant is corrupted by independent Gaussian noise. Notably this technique has been employed to obtain lower bounds on adaptive algorithms for sparse recovery in [PW12]. Even in the non-adaptive setting, Li and Woodruff [LW16] use the same hard distribution as we do to obtain sketching lower bounds for approximating Schatten p norms, the operator norm, and the Ky Fan norms. The technique to show their lower bounds is that if a sketching matrix has $k \leq c/(r^2 s^4)$ rows[3], it cannot distinguish between the random matrix $\boldsymbol{G}$ and the random matrix $\boldsymbol{G} + s\sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ where all the coordinates of $\boldsymbol{G}, \boldsymbol{u}_i, \boldsymbol{v}_i$ are drawn uniformly at random. They prove this by showing that $d_{\mathrm{TV}}(\mathscr{L}_1, \mathscr{L}_2)$ is small if $\mathscr{L}_1$ is the distribution of $M \cdot \mathrm{vec}(\boldsymbol{G})$

[2]Note that we assume a measurement matrix has orthonormal rows and that each measurement is corrupted by Gaussian noise of variance 1.

[3]Their lower bound is a bit more general, but we state this formulation for simplicity.

and $\mathscr{L}_2$ is the distribution of $M\,\mathrm{vec}(\boldsymbol{G} + s\sum_{i=1}^{n} \boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}})$ for any fixed measurement matrix M with $k \le c/(r^2 s^4)$ rows. Their techniques do not extend to the plant estimation in the distribution $\boldsymbol{G} + s\sum_{i=1}^{r} \boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}$ as the statement they prove only says that over the randomness of $\boldsymbol{G}, \boldsymbol{u}_i, \boldsymbol{v}_i$, the response distribution for $\boldsymbol{G} + s\sum_{i=1}^{r} \boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}$ is close to the response distribution for $\boldsymbol{G}$, but in our case, we want the distributions $\mathscr{L}_{u,v}$ and $\mathscr{L}_{u',v'}$ to be indistinguishable, where $\mathscr{L}_{u,v}$ is the response distribution for $\boldsymbol{G} + (\alpha/\sqrt{n})\sum_{i=1} u_i v_i^{\mathrm{T}}$.

Later, [SEAR18] considered the distribution of the symmetric random matrix $\boldsymbol{W} + \lambda \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}$, where $\boldsymbol{W}$ is the $n \times n$ symmetric Wigner matrix $(\boldsymbol{G} + \boldsymbol{G}^{\mathrm{T}})/\sqrt{2}$ and $\boldsymbol{U}$ is a uniformly random $n \times r$ matrix with orthonormal columns. They focus on obtaining lower bounds on adaptive algorithms that estimate the spike $\boldsymbol{U}$ in the matrix-vector product model. In particular, they show that if Q is a basis for the query space spanned by any deterministic algorithm after querying k adaptive matrix-vector queries, then $\lambda_r(Q^{\mathrm{T}}\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}Q)$ grows as $\sim \lambda^{k/r}$. Using this, they show that any algorithm which, given access to an arbitrary symmetric matrix A in the matrix-vector product query model, must use $\Omega(r\log(n)/\sqrt{\mathrm{gap}})$ adaptive queries to output an $n \times r$ orthonormal matrix V satisfying, for a small enough constant c,

$$\langle V, AV\rangle \ge (1 - c\cdot \mathrm{gap})\sum_{i=1}^{r} \lambda_i(A), \tag{5.2}$$

where $\mathrm{gap} = (\lambda_r(A) - \lambda_{r+1}(A))/\lambda_r(A)$. We note the above guarantee is non-standard in the low rank approximation literature, which instead focuses more on approximation algorithms for Frobenius norm and spectral norm LRA. While in the matrix-vector product model, their hard distribution helps in getting lower bounds for numerical linear algebra problems on symmetric matrices, it seems that our hard distribution $\boldsymbol{G} + (\alpha/\sqrt{n})\sum_{i=1}^{r} \boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}$ is easier to understand in the linear measurement model and gives the important property that the noise between rounds is independent, which is what lets us reduce the matrix-recovery problem with noisy measurements to spike estimation in $\boldsymbol{G} + (\alpha/\sqrt{n})\sum_{i=1}^{r} \boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}$.

Recently, Bakshi and Narayanan [BN23] obtained a tight lower bound for rank-1 spectral norm low rank approximation problem in the matrix-vector product model. They show that $\Omega(\log n/\sqrt{\varepsilon})$ matrix vector products are required to obtain a $1+\varepsilon$ spectral norm low rank approximation. We stress that while our results are not for $1+\varepsilon$ approximations, they hold in the stronger linear measurements model.

5.2 Preliminaries

5.2.1 Bayes Risk Lower Bounds

Let Θ be a set of parameters and $\mathcal{P} = \{ P_\theta : \theta \in \Theta \}$ be a set of distributions over $\mathcal{X}$. Let w be the prior distribution on Θ known to a learner. Suppose $\theta \sim w$, and $x \sim P_\theta$ and that the learner is given x. Now the learner wants to learn some information about θ. Let $\mathfrak{d} : \mathcal{X} \to \mathcal{A}$ be a mapping from sample x to action space $\mathcal{A}$ and $L : \Theta \times \mathcal{A} \to \{ 0, 1 \}$ be a zero-one loss function. We define

$$R_{\text{Bayes}}(L, \Theta, w) = \inf_{\mathfrak{d}} \int_\Theta \mathbf{E}_{x\sim P_\theta}[L(\theta, \mathfrak{d}(x))]w(\mathrm{d}\theta) = \inf_{\mathfrak{d}} E_{\theta\sim w}[\mathbf{E}_{x\sim P_\theta} L(\theta, \mathfrak{d}(x))],$$

and

$$R_0(L, \Theta, w) = \inf_{a\in\mathcal{A}} \int_\Theta L(\theta, a)w(\mathrm{d}\theta) = \inf_{a\in\mathcal{A}} \mathbf{E}_{\theta\sim w}[L(\theta, a)].$$

We drop Θ from the notation when it is clear.

f-divergence Let $\mathcal{C}$ denote the set of all convex functions f with $f(1) = 0$. Let P and Q be two distributions with densities p and q over a common measure μ. Then we have the f-divergence $D_f(P\|Q)$ defined as

$$D_f(P\|Q) = \int f\left(\frac{p}{q}\right) q\, \mathrm{d}\mu + f'(\infty)P\{ q = 0 \}$$

using the convention $0 \cdot \infty = \infty$. For $f = x\log(x)$, the f-divergence $D_f(P\|Q) = d_{\text{KL}}(P\|Q)$, the KullbackLeibler (KL) divergence. We frequently use $d_{\text{KL}}(\boldsymbol{X}\|\boldsymbol{Y})$ for some random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, which means the KL divergence between the distributions of $\boldsymbol{X}$ and $\boldsymbol{Y}$.

Now we can define the f-informativity of a set of distributions $\mathcal{P}$ with respect to a distribution w as follows:

$$I_f(\mathcal{P}, w) = \inf_Q \int D_f(P_\theta\|Q)w(\mathrm{d}\theta) = \inf_Q \mathbf{E}_{\theta\sim w}[D_f(P_\theta\|Q)].$$

We use $I(\mathcal{P}, w)$ to denote the f-informativity for $f = x\log x$. We finally have the following theorem from [CGZ16].

Theorem 5.2.1. *For any 0-1 loss function L,*

$$I_f(\mathcal{P}, w) \geq \phi_f(R_{\text{Bayes}}, R_0).$$

where for $0 \leq a, b \leq 1$, $\phi_f(a, b)$ denotes the f-divergence between distributions P, Q over $\{ 0, 1 \}$ with $P(1) = a$ and $Q(1) = b$.

As a corollary of the above theorem, [CGZ16] prove the following result which is the generalized Fano inequality.

Theorem 5.2.2. *For any prior w and 0-1 loss function L,*

$$R_{\text{Bayes}} \geq 1 + \frac{I(\mathscr{P}, w) + \log(1 + R_0)}{\log(1 - R_0)}.$$

5.2.2 KL Divergence

We state some properties of the KL-divergence that we use throughout this chapter.

Lemma 5.2.3. *Let P and Q be two random variables with probability measures p and q such that p is absolutely continuous with respect to q. Let $\widetilde{P}$, with probability measure $\widetilde{p}$, be the restriction of P to an event $\mathscr{E}$, i.e., for all $\mathscr{E}'$, $\mathbf{Pr}[\widetilde{P} \in \mathscr{E}'] = \mathbf{Pr}[P \in \mathscr{E} \cap \mathscr{E}']/\mathbf{Pr}(P \in \mathscr{E})$. Then*

$$d_{\text{KL}}(\widetilde{p} \| q) \leq \frac{1}{\mathbf{Pr}(P \in \mathscr{E})}(d_{\text{KL}}(p \| q) + 2).$$

Proof. For $x \in \mathscr{E}$, we have $\widetilde{p}(x) = p(x)/\mathbf{Pr}[P \in \mathscr{E}]$ and for $x \notin \mathscr{E}, \widetilde{p}(x) = 0$. By definition

$$d_{\text{KL}}(\widetilde{p} \| q) = \int \widetilde{p}(x) \log\left(\frac{\widetilde{p}(x)}{q(x)}\right) \mathrm{d}x$$

using the convention $0 \cdot \log(0/1) = 0$. Now,

$$\begin{aligned} d_{\text{KL}}(\widetilde{p} \| q) &= \int_{\mathscr{E}} \frac{p(x)}{\mathbf{Pr}(P \in \mathscr{E})} \log\left(\frac{p(x)}{\mathbf{Pr}(P \in \mathscr{E}) q(x)}\right) \mathrm{d}x \\ &= \frac{1}{\mathbf{Pr}(P \in \mathscr{E})} \int_{\mathscr{E}} p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x + \log(1/\mathbf{Pr}(P \in \mathscr{E})). \end{aligned}$$

We also have,

$$d_{\text{KL}}(p \| q) = \int_{\mathscr{E}} p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x + \int_{\bar{\mathscr{E}}} p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

which implies

$$\int_{\mathscr{E}} p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x = d_{\text{KL}}(p \| q) - \int_{\bar{\mathscr{E}}} p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x.$$

Thus, it is enough to lower bound $\int_{\bar{\mathscr{E}}} p(x) \log\left(p(x)/q(x)\right) \mathrm{d}x$ to obtain an upper bound on the left-hand side. So,

$$\int_{\bar{\mathscr{E}}} p(x)\log\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x = -\mathbf{Pr}(P\in\bar{\mathscr{E}})\int_{\bar{\mathscr{E}}}\frac{p(x)}{\mathbf{Pr}(P\in\bar{\mathscr{E}})}\log\left(\frac{q(x)}{p(x)}\right)\mathrm{d}x$$
$$\geq -\mathbf{Pr}(P\in\bar{\mathscr{E}})\log\left(\frac{\mathbf{Pr}(Q\in\bar{\mathscr{E}})}{\mathbf{Pr}(P\in\bar{\mathscr{E}})}\right)$$

where the last inequality follows from $-\log(x)$ being convex. Finally,

$$\begin{aligned}
d_{\mathrm{KL}}(\widetilde{p},q) &= \frac{1}{\mathbf{Pr}(P\in\mathscr{E})}\int_{x\in\mathscr{E}} p(x)\log\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x + \log(1/\mathbf{Pr}(P\in\mathscr{E}))\\
&\leq \frac{1}{\mathbf{Pr}(P\in\mathscr{E})}d_{\mathrm{KL}}(p,q) + \frac{\mathbf{Pr}(P\in\bar{\mathscr{E}})}{\mathbf{Pr}(P\in\mathscr{E})}\log(\mathbf{Pr}(Q\in\bar{\mathscr{E}})/\mathbf{Pr}(P\in\bar{\mathscr{E}})) + \log(1/\mathbf{Pr}(P\in\mathscr{E}))\\
&\leq \frac{1}{\mathbf{Pr}(P\in\mathscr{E})}d_{\mathrm{KL}}(p,q) + \frac{1}{\mathbf{Pr}(P\in\mathscr{E})} + \log(1/\mathbf{Pr}(P\in\mathscr{E}))
\end{aligned}$$

where we used $p\log(q/p) = p\log(q) + p\log(1/p) \leq 0 + 1 = 1$ for $0 \leq p, q \leq 1$. □

The following lemma states the chain rule for KL-divergence.

Lemma 5.2.4 (Chain Rule). *Let $(\boldsymbol{X}, \boldsymbol{Y})$ be jointly distributed random variables and $\boldsymbol{Z}, \boldsymbol{W}$ be independent random variables. Then*

$$d_{\mathrm{KL}}((\boldsymbol{X},\boldsymbol{Y})\|(\boldsymbol{Z},\boldsymbol{W})) = d_{\mathrm{KL}}(\boldsymbol{X}\|\boldsymbol{Z}) + \mathbf{E}_{\boldsymbol{X}}[d_{\mathrm{KL}}((\boldsymbol{Y}\mid\boldsymbol{X})\|\boldsymbol{W})].$$

The following lemma states the KL-divergence between two multivariate Gaussian distributions.

Lemma 5.2.5 (KL-divergence between Gaussians, folklore).

$$d_{\mathrm{KL}}(N(\mu_1,\Sigma_1)\|N(\mu_2,\Sigma_2)) = \frac{1}{2}[\log\frac{\det(\Sigma_2)}{\det(\Sigma_1)} + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^{\mathrm{T}}\Sigma_2^{-1}(\mu_2-\mu_1) - n]$$

where n is the dimension of the Gaussian.

In this article, we use the above lemma only for $\Sigma_1 = \Sigma_2 = I$ in which case the above lemma implies that $d_{\mathrm{KL}}(N(\mu_1, I)\|N(\mu_2, I)) = (1/2)\|\mu_2 - \mu_1\|_2^2$.

5.2.3 Properties of Gaussian random matrices and vectors

We use the following bounds on the norms of Gaussian matrices and vectors throughout: (i) If $\boldsymbol{g}$ is an n-dimensional Gaussian with all its entries independently drawn from $N(0,1)$, then with probability $\geq 1 - \exp(-\Theta(n))$, $(n/2) \leq \|\boldsymbol{g}\|_2^2 \leq (3n/2)$, and (ii) If $\boldsymbol{G}$ is an $m\times n$ ($m \geq n$) Gaussian matrix with i.i.d. entries from $N(0,1)$, then $\mathbf{Pr}[\|\boldsymbol{G}\|_2 \in [\sqrt{m}-\sqrt{n}-t, \sqrt{m}+\sqrt{n}+t]] \geq 1-\exp(-\Theta(t^2))$. See [RV09, LM00] and references therein for proofs.

Rotational Invariance We frequently use the rotational invariance property of multivariate Gaussian vectors $\boldsymbol{g} \sim N(0, I_n)$, which implies that for any $n \times n$ orthogonal matrix Q independent of $\boldsymbol{g}$, the vector $Q\boldsymbol{g}$ is also distributed as $N(0, I_n)$. Additionally, if $q_1, \dots, q_n$ denote rows of the matrix Q with q_1 chosen arbitrarily independent of $\boldsymbol{g}$ and the subsequent vectors

$$q_i \coloneqq f_i(q_1, \dots, q_{i-1}, \langle q_1, \boldsymbol{g}\rangle, \dots, \langle q_{i-1}, \boldsymbol{g}\rangle)$$

for arbitrary functions f_i satisfying $q_i \perp q_1, \dots, q_{i-1}$, then $\langle q_i, \boldsymbol{g}\rangle$ is distributed as $N(0, 1)$ and is independent of $\langle q_1, \boldsymbol{g}\rangle, \dots, \langle q_{i-1}, \boldsymbol{g}\rangle$. We crucially use this property in the proof of Theorem 5.3.4.

5.3 No. of Linear Measurements vs No. of Adaptive Rounds

We now state the main theorem which shows a lower bound on the number of adaptive rounds required for any deterministic algorithm to estimate the *plant* when the input is a random matrix $\boldsymbol{G} + (\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$. We use this theorem to prove Theorem 5.1.1 and lower bounds for other numerical linear algebra problems.

We prove the lower bound for the rank-r plant estimation by first proving a lower bound on the rank-1 plant estimation and then reducing the rank-1 recovery problem to rank-r recovery problem. For the rank-1 recovery problem, we prove the following lemma:

Lemma 5.3.1. *Given an integer n, and parameters $\alpha \geq 10$ and $\gamma \geq 1$, define the $n \times n$ random matrix $\boldsymbol{M} = \boldsymbol{G} + s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ for $s \coloneqq \alpha/\sqrt{n}$, where the entries of $\boldsymbol{G}, \boldsymbol{u}, \boldsymbol{v}$ are drawn independently from the distribution $N(0, 1)$. Let **Alg** be any t-round deterministic algorithm that makes $k \geq n$ adaptive linear measurements in each round. Let $Q^{(j)} \in \mathbb{R}^{k \times n^2}$ denote the matrix of general linear queries made by the algorithm in round j and Q be a matrix with orthonormal rows such that $\text{rowspan}(Q) = \text{rowspan}(Q^{(1)}) + \dots + \text{rowspan}(Q^{(t)})$. Then for all t such that $O(k \log(n)) \leq (K\alpha^2\gamma^2)^t k \leq O(n^2)$ for a universal constant K,*

$$\mathbf{Pr}_{\boldsymbol{M}=\boldsymbol{G}+s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 \leq (3K)k(K\alpha^2\gamma^2)^t] \geq (1 - 1/(10\gamma))^t - 1/\text{poly}(n).$$

Setting $\gamma = O(\log(n))$, the theorem shows that if $t \leq c\log(n^2/k)/(\log\log(n) + \log(\alpha))$ for a small enough constant c, then for any t-round deterministic algorithm,

$$\mathbf{Pr}_{\boldsymbol{M}=\boldsymbol{G}+s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 \leq n^2/100] \geq 4/5. \tag{5.3}$$

Setting $\gamma = O(1)$, we obtain that if $t \leq c\log(n^2/k)/(\log(\alpha))$ for a small enough constant c, then

$$\mathbf{Pr}_{\boldsymbol{M}=\boldsymbol{G}+s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 \leq n^2/100] \geq 1/\text{poly}(n). \tag{5.4}$$

The above lemma directly shows lower bounds on any deterministic algorithm that can approximate the rank-1 planted matrix.

5.3.1 Proof of Lemma 5.3.1

Before proceeding to the proof, we first discuss some notation, distributions and random variables that we use throughout the proof. Fix an arbitrary deterministic algorithm Alg. Let $A = G + suv^{\mathrm{T}}$ be a fixed realization of the random matrix $\boldsymbol{A}$. Recall $s = \alpha/\sqrt{n}$. The algorithm Alg queries a fixed orthonormal matrix $Q^{(1)}$ and receives the response $r^{(1)} \coloneqq Q^{(1)}\operatorname{vec}(A) = Q^{(1)}g + sQ^{(1)}(u \otimes v)$ where we use $g \coloneqq \operatorname{vec}(G)$. As Alg is *deterministic*, in the second round it queries the matrix $Q^{(2)}[r^{(1)}]$. We use $Q^{(2)}[r^{(1)}]$ to emphasize that Alg picks $Q^{(2)}$ purely as a function of $r^{(1)}$. It receives the response $r^{(2)} = Q^{(2)}[r^{(1)}]g + sQ^{(2)}[r^{(1)}](u \otimes v)$ and picks queries $Q^{(3)}[r^{(1)}, r^{(2)}]$ for the third round and so on. Using $h^{(j)} \coloneqq (r^{(1)}, \dots, r^{(j)})$ to denote the history of the algorithm until the end of round j, we have for $j = 0, \dots, t-1$

$$r^{(j+1)} \coloneqq Q^{(j+1)}[h^{(j)}](g + s \cdot u \otimes v).$$

For each $j = 1, \dots, t$, the randomness of $\boldsymbol{A}$ induces a distribution $H^{(j)}$ over histories until round j. Then for u and v, the distribution $H^{(j)}_{uv} \equiv H^{(j)} \mid \{\boldsymbol{u} = u, \boldsymbol{v} = v\}$ is the distribution of the history of Alg after j rounds conditioned on $\boldsymbol{u} = u$ and $\boldsymbol{v} = v$ where the randomness in the histories is purely from the randomness of $\boldsymbol{G}$. Recall that without loss of generality, we assume all the general linear queries made by the algorithm are orthogonal to each other. We first prove a tail bound on the amount of information that a non-adaptive query matrix can obtain.

A Tail Bound

Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be independent Gaussian random vectors. Let $Q \in \mathbb{R}^{k \times n^2}$ be an arbitrary matrix with k orthonormal rows. We have the following lemma.

Lemma 5.3.2. *There is a small enough universal constant β such that for all $k \geq n$ and C satisfying $4k \leq Ck \leq 16n^2$, we have for any orthonormal matrix $Q^{k \times n^2}$ that*

$$\mathbf{Pr}_{\boldsymbol{u},\boldsymbol{v}}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 \geq Ck] \leq \exp(-\beta Ck/n).$$

Proof. By Theorem 1.4 and Remark 1.5 of [Ver20], we have that

$$\mathbf{Pr}_{\boldsymbol{u},\boldsymbol{v}}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2 \geq \|Q\|_{\mathrm{F}} + t] \leq \exp\left(-\frac{ct^2}{2n}\right)$$

for all $0 \le t \le 2n$. For $t = \sqrt{Ck/4}$, for $4k \le Ck \le 16n^2$, we have that

$$\begin{aligned}\mathbf{Pr}_{\boldsymbol{u},\boldsymbol{v}}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2 \ge \sqrt{Ck}] &\le \mathbf{Pr}_{\boldsymbol{u},\boldsymbol{v}}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2 \ge \sqrt{k} + \sqrt{Ck/4}] \\ &\le \exp\left(-\frac{cCk}{8n}\right)\end{aligned}$$

which implies, taking $\beta = c/8$, that $\mathbf{Pr}_{\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{v}}}[\|Q(\widetilde{\boldsymbol{u}} \otimes \boldsymbol{v})\|_2^2 \ge Ck] \le \exp(-\beta Ck/n)$. □

First Round

Let $w^{(1)}$ be the distribution of $(\boldsymbol{u}, \boldsymbol{v})$ where $\boldsymbol{u}$ and $\boldsymbol{v}$ are random variables that are independently sampled from $N(0, I_n)$. As already defined, the distribution of the history of the algorithm after the first round for a fixed u, v is given by $H_{uv}^{(1)}$. We have that $H_{uv}^{(1)}$ is the distribution of the random variable

$$Q^{(1)}\boldsymbol{g} + sQ^{(1)}u \otimes v.$$

Let $P_{uv}^{(1)}$ be the distribution of the above random variable (although $P_{uv}^{(1)} \equiv H_{uv}^{(1)}$, we distinguish $P_{uv}^{(j)}$ and $H_{uv}^{(j)}$ in later rounds) and let $\mathscr{P}^{(1)} = \{ P_{uv}^{(1)} \}$ be the set of distributions for all u, v. Then we have

$$\begin{aligned}I(\mathscr{P}^{(1)}, w^{(1)}) &\le \mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(1)}}[d_{\mathrm{KL}}(Q^{(1)}\boldsymbol{g} + sQ^{(1)}(\boldsymbol{u} \otimes \boldsymbol{v}) \parallel \boldsymbol{g}_k)] \\ &\le s^2\, \mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(1)}} \|Q^{(1)}(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 = (\alpha^2/n)k.\end{aligned}$$

We define the loss function

$$\mathrm{Loss}^{(1)}((u, v), Q) = \begin{cases} 0 & \text{if } \|Q(u \otimes v)\|_2^2 \ge f_1(\alpha, \gamma)k \\ 1 & \text{if } \|Q(u \otimes v)\|_2^2 < f_1(\alpha, \gamma)k \end{cases}$$

for $Q \in \mathbb{R}^{k \times n^2}$ with orthonormal rows and some function $f_1(\alpha, \gamma)$ satisfying $4 \le f_1(\alpha, \gamma) \le 16n^2/k$ for a parameter $\gamma \ge 1$. From Lemma 5.3.2, we have

$$R_0(\mathrm{Loss}^{(1)}, w^{(1)}) \ge 1 - \exp(-\beta f_1(\alpha, \gamma)k/n)$$

which implies

$$R_{\mathrm{Bayes}} \ge 1 + \frac{(\alpha^2/n)k + \log(2)}{-\beta f_1(\alpha, \gamma)k/n + \log(2)}.$$

Picking $f_1(\alpha, \gamma) = K\alpha^2\gamma^2$, we have that $R_{\mathrm{Bayes}} \ge 1 - 1/(100\gamma^2)$. Thus, with probability $1 - 1/(10\gamma)$ over $(\boldsymbol{u}, \boldsymbol{v}) \sim w^{(1)}$, we have that

$$\mathbf{Pr}_{\boldsymbol{h}^{(1)} \sim P_{\boldsymbol{uv}}^{(1)}}[\|Q^{(2)}[\boldsymbol{h}^{(1)}](\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 \le f_1(\alpha, \gamma)k] \ge 1 - 1/(10\gamma).$$

Let $\text{Good}^{(1)}$ be the set of all (u, v) that satisfy the above property. Let $w^{(2)}$ be the distribution of $(\boldsymbol{u}, \boldsymbol{v}) \sim w^{(1)}$ conditioned on $(\boldsymbol{u}, \boldsymbol{v}) \in \text{Good}^{(1)}$. For each $(u, v) \in \text{Good}^{(1)}$, let

$$\text{GoodH}_{uv}^{(1)} = \{\, h^{(1)} \ : \ \|Q^{(2)}[h^{(1)}](u \otimes v)\|_2^2 \le f_1(\alpha, \gamma)k \,\} .$$

We have for all $(u, v) \in \text{Good}^{(1)}$ that $\mathbf{Pr}_{\boldsymbol{h}^{(1)} \sim P_{uv}^{(1)}}[\boldsymbol{h}^{(1)} \in \text{GoodH}_{uv}^{(1)}] \ge 1 - 1/(10\gamma)$. The overall conclusion of this is that with a large probability over $(\boldsymbol{u}, \boldsymbol{v}) \sim w^{(1)}$, the squared projection of $\boldsymbol{u} \otimes \boldsymbol{v}$ on the query asked by Alg in round 2 is small with large probability over $\boldsymbol{G}$.

Further Rounds

We first define the following objects inductively.

1. For $j \ge 2$, let $w^{(j)}$ be the distribution of $(\boldsymbol{u}, \boldsymbol{v}) \sim w^{(j-1)}$ conditioned on $(\boldsymbol{u}, \boldsymbol{v}) \in \text{Good}^{(j-1)}$. So $w^{(j)}$ is the distribution over only those inputs for which the squared projection of $u \otimes v$ on the query space of Alg until round j is small with high probability over $\boldsymbol{G}$.
2. For $(u, v) \in \text{Good}^{(j-1)}$,

$$P_{uv}^{(j)} \coloneqq \text{ distribution of } \boldsymbol{h}^{(j)} = (\boldsymbol{h}^{(j-1)}, \boldsymbol{r}^{(j)}) \sim \boldsymbol{H}_{uv}^{(j)} \text{ conditioned on } \boldsymbol{h}^{(j-1)} \in \text{GoodH}_{uv}^{(j-1)}.$$

   $P_{uv}^{(j)}$ denotes the distribution over histories after round j, conditioned on the queries used until round j not having a lot of "information" about $u \otimes v$.
3. Let $\mathscr{P}^{(j)} \coloneqq \{\, P_{uv}^{(j)} \ : \ (u, v) \in \text{Good}^{(j-1)} \,\}$.
4. For $j \ge 1$,

$$\text{Good}^{(j)} \coloneqq$$
$$\{\, (u, v) \in \text{Good}^{(j-1)} \ : \ \mathbf{Pr}_{\boldsymbol{h}^{(j)} \sim P_{uv}^{(j)}}[\|Q^{(j+1)}[\boldsymbol{h}^{(j)}](u \otimes v)\|_2^2 \le f_j(\alpha, \gamma)k] \ge 1 - 1/(10\gamma) \,\} .$$

   The set $\text{Good}^{(j)}$ denotes those values of (u, v) for which with large probability over $\boldsymbol{G}$, the queries used by the algorithm until round $j+1$ do not have a lot of "information" about $u \otimes v$.
5. For $(u, v) \in \text{Good}^{(j)}$,

$$\text{GoodH}_{uv}^{(j)} \coloneqq$$
$$\{\, h^{(j)} = (h^{(j-1)}, r^{(j)}) \ : \ h^{(j-1)} \in \text{GoodH}_{uv}^{(j-1)} \text{ and } \|Q^{(j+1)}[h^{(j)}](u \otimes v)\|_2^2 \le f_j(\alpha, \gamma)k \,\} .$$

   $\text{GoodH}_{uv}^{(j)}$ denotes those histories, for which the queries used by the algorithm until round $j+1$ do not have a lot of "information" about $u \otimes v$.
6. Let $f_0(\alpha, \gamma) = K$ and for $j \ge 1$, let $f_j(\alpha, \gamma) = K\alpha^2\gamma^2 f_{j-1}(\alpha, \gamma)$ for a large enough universal constant K.

Lemma 5.3.3. *For all $1 \le j$ satisfying $f_j(\alpha,\gamma)k \le 16n^2$,*

1. $$\mathbf{Pr}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[(\boldsymbol{u},\boldsymbol{v}) \in \mathsf{Good}^{(j)}] \ge 1 - \frac{1}{10\gamma}$$

2. *For all $(u,v) \in \mathsf{Good}^{(j)}$,*
$$\mathbf{Pr}_{\boldsymbol{h}^{(j)}\sim P_{uv}^{(j)}}[\boldsymbol{h}^{(j)} \in \mathsf{GoodH}_{uv}^{(j)}] \ge 1 - \frac{1}{10\gamma}$$

3. $$\mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[d_{\mathrm{KL}}(P_{\boldsymbol{uv}}^{(j)} \,\|\, G^{(j)})] \le f_j(\alpha,\gamma)(k/n),$$
where $G^{(j)}$ is the joint distribution of j independent *k-dimensional Gaussian random variables.*

Proof. From the previous section, all the above statements hold for $j = 1$. Now assume that all the above statements hold for rounds $1,\dots,j-1$. We prove the statements inductively for round j. Recall $w^{(j)}$ is defined to be the distribution of $(\boldsymbol{u},\boldsymbol{v}) \sim w^{(j-1)}$ conditioned on $(\boldsymbol{u},\boldsymbol{v}) \in \mathsf{Good}^{(j-1)}$. For each $(u,v) \in \mathsf{Good}^{(j-1)}$, we now bound $d(P_{uv}^{(j)}\|G^{(j)})$. By definition,

$$d(P_{uv}^{(j)}\|G^{(j)}) = d_{\mathrm{KL}}((\boldsymbol{h}^{(j-1)},\boldsymbol{r}^{(j)})\|(\boldsymbol{g}_k^{(1)},\dots,\boldsymbol{g}_k^{(j)}))$$

where $\boldsymbol{h}^{(j)} = (\boldsymbol{h}^{(j-1)},\boldsymbol{r}^{(j)}) \sim P_{uv}^{(j)}$. Note that the marginal distribution of $\boldsymbol{h}^{(j-1)}$ is given by conditioning the distribution $P_{uv}^{(j-1)}$ on the event $\mathsf{GoodH}_{uv}^{(j-1)}$. By Lemma 5.2.4, we have

$$\begin{aligned}
&d_{\mathrm{KL}}((\boldsymbol{h}^{(j-1)},\boldsymbol{r}^{(j)})\|(\boldsymbol{g}_k^{(1)},\dots,\boldsymbol{g}_k^{(j)}))\\
&= d_{\mathrm{KL}}(\boldsymbol{h}^{(j-1)}\|(\boldsymbol{g}_k^{(1)},\dots,\boldsymbol{g}_k^{(j-1)})) + \mathbf{E}_{\boldsymbol{h}^{(j-1)}}[d_{\mathrm{KL}}((\boldsymbol{r}^{(j)} \mid \boldsymbol{h}^{(j-1)})\|\boldsymbol{g}_k^{(j)}]\\
&= d_{\mathrm{KL}}((P_{uv}^{(j-1)} \mid \mathsf{GoodH}_{uv}^{(j-1)})\|G^{(j-1)}) + \mathbf{E}_{\boldsymbol{h}^{(j-1)}\sim P_{uv}^{(j-1)}|\mathsf{GoodH}_{uv}^{(j-1)}}[d_{\mathrm{KL}}((\boldsymbol{r}^{(j)} \mid \boldsymbol{h}^{(j-1)})\|\boldsymbol{g}_k^{(j)}]. \qquad (5.5)
\end{aligned}$$

Now, using Lemma 5.2.3, we have

$$\begin{aligned}
d_{\mathrm{KL}}((P_{uv}^{(j-1)} \mid \mathsf{GoodH}_{uv}^{(j-1)})\|G^{(j-1)}) &\le \frac{d_{\mathrm{KL}}(P_{uv}^{(j-1)}\|G^{(j-1)}) + 2}{\mathbf{Pr}_{\boldsymbol{h}^{(j-1)}\sim P_{uv}^{(j-1)}}[\boldsymbol{h}^{(j-1)} \in \mathsf{GoodH}^{(j-1)}]}\\
&\le (5/4)d_{\mathrm{KL}}(P_{uv}^{(j-1)}\|G^{(j-1)}) + 5/2.
\end{aligned}$$

Here we used the inductive assumption that $\mathbf{Pr}_{\boldsymbol{h}^{(j-1)}\sim P_{uv}^{(j-1)}}[\boldsymbol{h}^{(j-1)} \in \mathsf{GoodH}^{(j-1)}] \ge 1 - 1/(10\gamma) \ge 9/10$ where the last inequality follows as the parameter $\gamma \ge 1$. Next, we upper bound the second term in (5.5). We have that $\boldsymbol{r}^{(j)}|\boldsymbol{h}^{(j-1)} = Q^{(j)}[\boldsymbol{h}^{(j-1)}]\boldsymbol{g} + s(Q^{(j)}[\boldsymbol{h}^{(j-1)}])(u \otimes v)$ is distributed as $N(s(Q^{(j)}[\boldsymbol{h}^{(j-1)}])(u \otimes v), I_k)$ by rotational invariance of the Gaussian distribution. Therefore,

$$d_{\mathrm{KL}}((\boldsymbol{r}^{(j)}|\boldsymbol{h}^{(j-1)})\|\boldsymbol{g}_k^{(j)}) = (1/2)s^2\|Q^{(j)}[\boldsymbol{h}^{(j-1)}](u \otimes v)\|_2^2 \le (\alpha^2/n)f_{j-1}(\alpha,\gamma)k.$$

In the last inequality, we used the fact that $\boldsymbol{h}^{(j-1)} \in \mathrm{GoodH}_{uv}^{(j-1)}$. Finally,

$$\begin{aligned}\mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[d(P_{\boldsymbol{uv}}^{(j)}\|G^{(j)})] &\le (5/4)\,\mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}\, d_{\mathrm{KL}}(P_{\boldsymbol{uv}}^{(j-1)}\|G^{(j-1)}) + 5/2 + \alpha^2 f_{j-1}(\alpha,\gamma)(k/n)\\ &\le (5/2)\,\mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j-1)}}\, d_{\mathrm{KL}}(P_{\boldsymbol{uv}}^{(j-1)}\|G^{(j-1)}) + 5/2 + \alpha^2 f_{j-1}(\alpha,\gamma)(k/n)\end{aligned}$$

as the distribution $w^{(j)}$ is obtained by conditioning $w^{(j-1)}$ on an event with probability $\ge 1 - 1/(10\gamma) \ge 1/2$ and $d_{\mathrm{KL}}(\cdot\|\cdot) \ge 0$. Now, using the inductive assumption, we have

$$\begin{aligned}\mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[d(P_{\boldsymbol{uv}}^{(j)}\|G^{(j)})] &\le (5/2)f_{j-1}(\alpha,\gamma)(k/n) + 5/2 + \alpha^2 f_{j-1}(\alpha,\gamma)(k/n)\\ &\le 2\alpha^2 f_{j-1}(\alpha,\gamma)(k/n) \le f_j(\alpha,\gamma)(k/n)\end{aligned}$$

where we use $\alpha \ge 15$. This proves the third statement in the lemma for round j. We also have, $I(w^{(j)}, \mathscr{P}^{(j)}) \le \mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[d(P_{\boldsymbol{uv}}^{(j)}\|G^{(j)})] \le 2\alpha^2 f_{j-1}(\alpha,\gamma)(k/n)$. We now define a loss function $L^{(j)}$ and use Bayes risk lower bounds to prove the remaining statements. Let

$$\mathrm{Loss}^{(j)}((u,v),Q) = \begin{cases} 1 & \text{if } \|Q(u\otimes v)\|_2^2 \le f_j(\alpha,\gamma)k\\ 0 & \text{if } \|Q(u\otimes v)\|_2^2 > f_j(\alpha,\gamma)k\end{cases}$$

where $Q \in \mathbb{R}^{k\times n^2}$ is an orthonormal matrix with $k \ge n$ rows. We have for j such that $f_j(\alpha,\gamma)k \le 16n^2$,

$$R_0(\mathrm{Loss}^{(j)}, w^{(j)}) = \inf_Q \mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[\mathrm{Loss}^{(j)}((\boldsymbol{u},\boldsymbol{v}),Q)] \ge 1-(1-1/(10\gamma))^{-(j-1)}\exp(-\beta f_j(\alpha,\gamma)k/n)$$

where we use Lemma 5.3.2 and the fact that the distribution $w^{(j)}$ is obtained by conditioning $w^{(1)}$ on an event with probability $\ge (1-1/(10\gamma))^{j-1}$, which is obtained by chaining the first induction hypothesis. By the generalized Fano inequality, we obtain

$$\begin{aligned}R_{\mathrm{Bayes}}(\mathrm{Loss}^{(j)}, w^{(j)}) &\ge 1 + \frac{I(w^{(j)},\mathscr{P}^{(j)}) + \log(2)}{\log(1 - R_0(\mathrm{Loss}^{(j)}, w^{(j)}))}\\ &\ge 1 - \frac{2\alpha^2 f_{j-1}(\alpha,\gamma)k/n + \log(2)}{\beta f_j(\alpha,\gamma)k/n + (j-1)\log(1-1/(10\gamma))}.\end{aligned}$$

As $f_j(\alpha,\gamma) = K\alpha^2\gamma^2 f_{j-1}(\alpha,\gamma)$ and $f_0(\alpha,\gamma) \ge K$ for a large enough constant K, we have

$$\beta f_j(\alpha,\gamma) \ge -10j\log(1-1/(10\gamma))$$

and therefore for K large enough,

$$R_{\text{Bayes}}(\text{Loss}^{(j)}, w^{(j)}) \geq 1 - \frac{1}{100\gamma^2}.$$

By definition of R_{Bayes}, we conclude that

$$\mathbf{E}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[\mathbf{E}_{\boldsymbol{h}^{(j)}\sim P_{\boldsymbol{uv}}^{(j)}}[\text{Loss}((\boldsymbol{u},\boldsymbol{v}), Q^{(j+1)}[\boldsymbol{h}^{(j)}])]] \geq 1 - 1/(100\gamma^2).$$

By Markov's inequality, we have that with probability $\geq 1-1/(10\gamma)$ over $(\boldsymbol{u},\boldsymbol{v}) \sim w^{(j)}$, it holds that

$$\mathbf{E}_{\boldsymbol{h}^{(j)}\sim P_{\boldsymbol{uv}}^{(j)}}[\text{Loss}((\boldsymbol{u},\boldsymbol{v}), Q^{(j+1)}[\boldsymbol{h}^{(j)}])] \geq 1 - 1/(10\gamma)$$

which is equivalent to

$$\mathbf{Pr}_{\boldsymbol{h}^{(j)}\sim P_{\boldsymbol{uv}}^{(j)}}[\|Q^{(j+1)}[\boldsymbol{h}^{(j)}](\boldsymbol{u}\otimes\boldsymbol{v})\|_2^2 \leq f_j(\alpha,\gamma)k] \geq 1 - 1/(10\gamma).$$

Thus, we conclude that $\mathbf{Pr}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[(\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j)}] \geq 1 - 1/(10\gamma)$ and that for $(u, v) \in \text{Good}^{(j)}$, we have $\mathbf{Pr}_{\boldsymbol{h}^{(j)}\sim P_{uv}^{(j)}}[\boldsymbol{h}^{(j)} \in \text{GoodH}_{uv}^{(j)}] \geq 1 - 1/(10\gamma)$. □

Wrap-up

Let $j \geq 1$ satisfy $f_j(\alpha,\gamma)k \leq 16n^2$. By definition, $\text{Good}^{(j)} \subseteq \text{Good}^{(j-1)} \subseteq \ldots \text{Good}^{(1)}$. We also have

$$w^{(j)} = w^{(j-1)} \mid \text{Good}^{(j-1)} = w^{(1)} \mid \text{Good}^{(j-1)}.$$

Now, using the fact that $\text{Good}^{(j)} \subseteq \text{Good}^{(j-1)}$,

$$\frac{\mathbf{Pr}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(1)}}[(\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j)}]}{\mathbf{Pr}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(1)}}[(\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j-1)}]} = \mathbf{Pr}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(1)}}[(\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j)} \mid (\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j-1)}]$$

As $w^{(j)} = w^{(1)} \mid \text{Good}^{(j-1)}$, we have

$$\begin{aligned}\mathbf{Pr}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(1)}}[(\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j)} \mid (\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j-1)}] &= \mathbf{Pr}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(j)}}[(\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j)}] \\ &\geq (1 - 1/(10\gamma))\end{aligned}$$

where the last inequality is from Lemma 5.3.3. Thus, $\mathbf{Pr}_{(\boldsymbol{u},\boldsymbol{v})\sim w^{(1)}}[(\boldsymbol{u},\boldsymbol{v}) \in \text{Good}^{(j)}] \geq (1-1/(10\gamma))^j$.

Similarly, for $(u, v) \in \text{Good}^{(j)}$, we have that

$$\frac{\mathbf{Pr}_{\boldsymbol{h}^{(j)} \sim H_{uv}^{(j)}}[\boldsymbol{h}^{(j)} \in \text{GoodH}_{uv}^{(j)}]}{\mathbf{Pr}_{\boldsymbol{h}^{(j-1)} \sim H_{uv}^{(j-1)}}[\boldsymbol{h}^{(j-1)} \in \text{GoodH}_{uv}^{(j-1)}]} = \mathbf{Pr}_{\boldsymbol{h}^{(j)} \sim P_{uv}^{(j)}}[\boldsymbol{h}^{(j)} \in \text{GoodH}_{uv}^{(j)}]$$
$$\geq (1 - 1/(10\gamma))$$

where again, the last inequality follows from Lemma 5.3.3. Thus,

$$\mathbf{Pr}_{\boldsymbol{h}^{(j)} \sim H_{uv}^{(j)}}[\boldsymbol{h}^{(j)} \in \text{GoodH}_{uv}^{(j)}] \geq (1 - 1/(10\gamma))^j.$$

Now, for $(\boldsymbol{u}, \boldsymbol{v}) \sim w^{(1)}$ and $\boldsymbol{h}^{(j)} \sim H_{\boldsymbol{uv}}^{(j)}$, we have that with probability $\geq (1 - 1/(10\gamma))^{2j}$ it holds that $(\boldsymbol{u}, \boldsymbol{v}) \in \text{Good}^{(j)}$ and $\boldsymbol{h}^{(j)} \in \text{GoodH}_{\boldsymbol{uv}}^{(j)}$. Thus, with probability $\geq (1 - 1/(10\gamma))^{2j}$ over the input matrix $\boldsymbol{G} + \alpha \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$, we have that

$$\sum_{j'=1}^{j} \|Q^{(j'+1)}[\boldsymbol{h}^{(j')}](\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 \leq \sum_{j'=1}^{j} f_{j'}(\alpha, \gamma)k \leq 2f_j(\alpha, \gamma)k.$$

With probability $\geq 1 - 1/\text{poly}(n)$, using Lemma 5.3.2, $\|Q^{(1)}(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 \leq k \log(n)$. Using a union bound, we obtain that with probability $\geq (1 - 1/(10\gamma))^{2j} - 1/\text{poly}(n)$,

$$\begin{aligned}\sum_{j'=0}^{j} \|Q^{(j'+1)}[\boldsymbol{h}^{(j')}](\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 &\leq 2f_j(\alpha, \gamma)kr + Kk \log(n) \\ &= 2kf_0(\alpha, \gamma)(K\alpha^2\gamma^2)^j + Kk \log(n) \\ &= (3K)k(K\alpha^2\gamma^2)^j\end{aligned}$$

where K is a large enough absolute constant which proves the lemma for j such that $(K\alpha^2\gamma^2)^j = \Omega(\log(n))$.

5.3.2 Lower Bounds for estimating rank-r Plant

We show that the lower bounds on number of linear measurements required to estimate the rank-1 planted matrix can be extended to algorithms that estimate the rank-r planted matrix as well.

Theorem 5.3.4. *Let n and $r \leq n/2$ be input parameters and α be a large enough constant. Let the random matrix $\boldsymbol{G} + (\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ be the input which can be accessed using linear measurements. If* **Alg** *is a t-round adaptive algorithm that uses k linear measurements and at the end of t-rounds outputs a matrix $\hat{A}$ such* *that with probability* $\geq 9/10$ *over the randomness of the input and the internal randomness of the algorithm,*

$$\|\hat{A} - \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \leq c(n^2 r)$$

for a small enough constant c, then $t = \Omega(\log(n^2/k)/(\log(\alpha) + \log\log n))$.

Proof. Suppose Alg is a t round algorithm that uses k liner measurements in each round such that when run on the matrix $\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$, it produces a matrix $\hat{A}$ such that with probability $9/10$ over the input matrix,

$$\|\hat{A} - \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\|_{\mathrm{F}}^2 \leq cn^2$$

for a small enough constant c. By making the algorithm to query the vector $\mathrm{vec}(\hat{A})$ in round $t+1$, we can therefore ensure that if Q is the overall query space of the algorithm, then

$$\mathbf{Pr}[\|Q(\boldsymbol{u} \otimes \boldsymbol{v})\|_2^2 \geq n^2/100] \geq 4/5.$$

By Lemma 5.3.1, we obtain that $t + 1 = \Omega(\log(n^2/k)/(\log(\alpha) + \log\log n))$ and therefore

$$t = \Omega(\log(n^2/k)/(\log(\alpha) + \log\log n)).$$

Now suppose Alg_r is a t round algorithm that uses k linear measurements of the input random matrix $\boldsymbol{G} + (\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ and outputs a matrix $\hat{A}$ such that with probability $\geq 9/10$ over the input,

$$\|\hat{A} - \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \leq cn^2 r$$

for a small enough constant c. We obtain a lower bound on t by reducing the rank-1 plant estimation problem to the rank-r plant estimation problem. Suppose the input is the random matrix $\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}$. We sample $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_r$ and $\boldsymbol{v}_2, \ldots, \boldsymbol{v}_r$ so that the coordinates of these vectors are independent standard Gaussian random variables. Now we note that we can perform arbitrary linear measurements of the matrix $\boldsymbol{G} + (\alpha/\sqrt{n}) \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ since we have access to linear measurements of $\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}$ and we know the vectors $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_r$ and $\boldsymbol{v}_2, \ldots, \boldsymbol{v}_r$. We can then use Algorithm Alg_r to obtain a matrix $\hat{A}$ such that with probability $\geq 9/10$ over the input and our sampled vectors,

$$\|\sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} - \hat{A}\|_{\mathrm{F}}^2 \leq cn^2 r.$$

Condition on this event. We have $\|\hat{A} - [\hat{A}]_r\|_{\mathrm{F}}^2 \leq \|\sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} - \hat{A}\|_{\mathrm{F}}^2 \leq cn^2 r$ which then implies

$\| \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} - [\hat{A}]_r\|_{\mathrm{F}}^2 \le 4cn^2r$ using the triangular inequality. Now, let P be the at most rank r projection matrix onto the rowspace of the matrix $[\hat{A}]_r$ and assume that $\boldsymbol{U}$ and $\boldsymbol{V}$ are matrices with columns given by $\boldsymbol{u}_1, \dots, \boldsymbol{u}_r$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_r$ respectively. From above, we get

$$\|\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}(I - P)\|_{\mathrm{F}}^2 \le \|\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}} - [\hat{A}]_r\|_{\mathrm{F}}^2 \le 4cn^2r$$

which then implies that

$$\sigma_{\min}(\boldsymbol{U})^2\|\boldsymbol{V}^{\mathrm{T}}(I - P)\|_{\mathrm{F}}^2 \le \|\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}(I - P)\|_{\mathrm{F}}^2 \le 4cn^2r.$$

Now, $\boldsymbol{U}$ is an $n \times r$ matrix with coordinates being independent standard Gaussian random variables. We have from [RV09] that if $r \le n/2$, then with probability $1 - \exp(-n)$, $\sigma_{\min}(\boldsymbol{U}) \ge c_1\sqrt{n}$. From [LM00] we also have that with probability $\ge 1 - \exp(-n)$, $c_2 n \le \|\boldsymbol{u}_1\|_2^2, \dots, \|\boldsymbol{u}_r\|_2^2 \le Cn$ for a small enough c_2 and large enough C. Conditioned on all these events, we have

$$\begin{aligned}\sum_{i=1}^{r} \|\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}(I - P)\|_{\mathrm{F}}^2 &\le Cn\sum_{i=1}^{r}\|\boldsymbol{v}_i^{\mathrm{T}}(I - P)\|_2^2 \\ &\le Cn\|\boldsymbol{V}^{\mathrm{T}}(I - P)\|_{\mathrm{F}}^2 \le \frac{(4cn^2r)(Cn)}{c_1^2 n} \le (4cC/c_1)n^2.\end{aligned}$$

Hence, at least $9r/10$ indices i have the property that $\|\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}(I - P)\|_{\mathrm{F}}^2 \le (40cC/c_1)n^2$. Since the marginal distributions of $\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}, \dots, \boldsymbol{u}_r\boldsymbol{v}_r^{\mathrm{T}}$ are identical, we obtain that with probability $\ge 8/10$,

$$\|\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}(I - P)\|_{\mathrm{F}}^2 \le (40cC/c_1)n^2.$$

Note that rank of P is at most r and therefore $P = QQ^{\mathrm{T}}$ for an orthonormal matrix Q with at most r columns. In the $(t+1)$-th round, we make nr linear measurements of the input matrix and obtain the matrix

$$\boldsymbol{G}Q + (\alpha/\sqrt{n})\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}Q$$

and therefore, we can obtain the matrix $M = \boldsymbol{G}P + (\alpha/\sqrt{n})\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}P$. Let $[M]_1$ be the best rank-1 approximation of M in operator norm. We have

$$\begin{aligned}\|[M]_1 - (\alpha/\sqrt{n})\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}\|_2 &\le \|[M]_1 - M\|_2 + \|M - (\alpha/\sqrt{n})\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}P\|_2 \\ &\quad + \|(\alpha/\sqrt{n})\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}P - (\alpha/\sqrt{n})\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}\|_2 \\ &\le \|\boldsymbol{G}P\|_2 + \|\boldsymbol{G}P\|_2 + (\alpha/\sqrt{n})\sqrt{(40cC/c_1)n^2}.\end{aligned}$$

Since, $\|G\|_2 \le 2\sqrt{n}$ with probability $1 - \exp(-\Theta(n))$, we get

$$\|(\sqrt{n}/\alpha)[M]_1 - \boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}\|_2 \le (4/\alpha + \sqrt{40cC/c_1})n$$

and

$$\|(\sqrt{n}/\alpha)[M]_1 - \boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}\|_{\mathrm{F}}^2 \le 2(4/\alpha + \sqrt{40cC/c_1})^2 n^2.$$

For α large enough and c small enough, by querying the vector $\mathrm{vec}([M]_1)$, we can again ensure that if Q is the query space of the algorithm after $t+2$ rounds, we have with probability $\ge 1/2$ over the input random matrix that

$$\|Q(\boldsymbol{u}_1 \otimes \boldsymbol{v}_1)\|_2^2 \ge n^2/100.$$

We again have from Lemma 5.3.1 that $t + 2 = \Omega(\log(n^2/k)/(\log\log n + \log\alpha))$ and therefore that $t = \Omega(\log(n^2/k)/(\log\log n + \log\alpha))$. □

5.4 Proof of Theorem 5.1.1

Using the reduction from matrix recovery with noisy measurements to plant estimation with exact measurements that we described in the previous sections, we can now prove Theorem 5.1.1.

Proof of Theorem 5.1.1. Let $\mathscr{A}$ be any randomized algorithm that queries orthonormal measurements of the underlying matrix A and outputs a reconstruction $\hat{A}$ of A satisfying $\|A - \hat{A}\|_{\mathrm{F}}^2 \le c\|A\|_{\mathrm{F}}^2$ with probability $\ge p$, over both the randomness of the algorithm and randomness of the measurements. Let $\hat{A} = \mathscr{A}(A, \boldsymbol{\sigma}, \boldsymbol{\gamma})$ where $\boldsymbol{\sigma}$ captures the randomness of the algorithm and $\boldsymbol{\gamma}$ captures the randomness in measurements. First we have the following lemma.

Lemma 5.4.1. *If there is a randomized algorithm $\mathscr{A}(A, \boldsymbol{\sigma}, \boldsymbol{\gamma})$ such that for all rank $\le r$ matrices A with $\|A\|_{\mathrm{F}}^2 = \Theta(nr)$,*

$$\mathbf{Pr}_{\boldsymbol{\sigma},\boldsymbol{\gamma}}[\|\mathscr{A}(A, \boldsymbol{\sigma}, \boldsymbol{\gamma}) - A\|_{\mathrm{F}}^2 \le c\|A\|_{\mathrm{F}}^2] \ge p,$$

then there is a randomized algorithm $\mathscr{A}'$ such that for all A with $\|A\|_{\mathrm{F}}^2 = \Theta(nr)$,

$$\mathbf{Pr}_{G,\boldsymbol{\sigma}}[\|\mathscr{A}'(A + G, \boldsymbol{\sigma}) - A\|_{\mathrm{F}}^2 \le c\|A\|_{\mathrm{F}}^2] \ge p.$$

Proof. Fix a particular σ. The algorithm first queries an orthonormal matrix $Q^{(1)} \in \mathbb{R}^{k\times n^2}$ and receives a random response $\boldsymbol{r}^{(1)} = Q^{(1)} \cdot \mathrm{vec}(A) + \boldsymbol{g}^{(1)}$ where $\boldsymbol{g}^{(1)}$ is a vector with independent Gaussian components of mean 0 and variance 1. As a function of $\boldsymbol{r}^{(1)}$, the algorithm queries an orthonormal matrix $Q^{(1)}[\boldsymbol{r}^{(1)}]$ and receives the random response $\boldsymbol{r}^{(2)} = Q^{(2)}[\boldsymbol{r}^{(1)}] \cdot \mathrm{vec}(A) + \boldsymbol{g}^{(2)}$ where $\boldsymbol{g}^{(2)}$ is again a random Gaussian vector *independent* of $\boldsymbol{g}^{(1)}$. Importantly, we also have that $Q^{(2)}[\boldsymbol{r}^{(1)}]$ is orthogonal to the query matrix $Q^{(1)}$ in the first round. The algorithm proceeds in further rounds accordingly where it picks query matrices as a function of the responses in all the previous rounds.

Let $\boldsymbol{G}$ be an $n \times n$ Gaussian matrix with independent entries of mean 0 and variance 1. We now observe that $\boldsymbol{r}^{(1)} = Q^{(1)} \cdot \mathrm{vec}(A) + \boldsymbol{g}^{(1)}$ has the same distribution as $Q^{(1)} \cdot \mathrm{vec}(A + \boldsymbol{G})$ by rotational invariance of Gaussian distribution. Now conditioning on $\boldsymbol{r}^{(1)}$, we obtain that $\boldsymbol{r}^{(2)} = Q^{(2)}[\boldsymbol{r}^{(1)}] \cdot \mathrm{vec}(A) + \boldsymbol{g}^{(2)}$ has the *same* distribution as $Q^{(2)}[\boldsymbol{r}^{(1)}] \cdot \mathrm{vec}(A + \boldsymbol{G})$ as $Q^{(2)}[\boldsymbol{r}^{(1)}]$ is orthogonal to $Q^{(1)}$ and hence $Q^{(2)}[\boldsymbol{r}^{(1)}] \cdot \boldsymbol{g}$ is independent of $Q^{(1)}\boldsymbol{g}$, again by rotational invariance.

Thus, we can consider a deterministic algorithm $\mathcal{A}'$ parameterized by the same σ which exactly simulates the behavior of $\mathcal{A}$ performing matrix recovery with noisy measurements. Thus,

$$\mathbf{Pr}_{\boldsymbol{G}}[\|\mathcal{A}'(A + \boldsymbol{G}, \sigma) - A\|_{\mathrm{F}}^2 \le c\|A\|_{\mathrm{F}}^2] = \mathbf{Pr}_{\boldsymbol{\gamma}}[\|\mathcal{A}(A, \sigma, \boldsymbol{\gamma}) - A\|_{\mathrm{F}}^2 \le c\|A\|_{\mathrm{F}}^2].$$

Taking expectation over $\boldsymbol{\sigma}$, we obtain the result. □

Let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ be $2r$ random vectors with coordinates being independent standard normal random variables. Let $\boldsymbol{U}$ be an $n \times r$ matrix with columns given by $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$ and $\boldsymbol{V}$ be an $n \times r$ matrix with columns given by $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$.

Claim 5.4.2. *If $r \le n/2$, With high probability,*

$$\|\sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2 = \Theta(n^2 r).$$

Proof. By definition, $\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}} = \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$. We now have $2nr \ge \|\boldsymbol{V}\|_{\mathrm{F}}^2 \ge nr/2$ with probability $1 - \exp(-\Theta(nr))$ from [LM00]. From [RV09], with probability $\ge 1 - \exp(-\Theta(n))$, $\sigma_{\min}(\boldsymbol{U}) \ge c'\sqrt{n}$ for a constant c' and as seen in preliminaries, $\|\boldsymbol{U}\|_2 \le 2\sqrt{n}$. Thus, conditioned on both these events,

$$\|\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}\|_{\mathrm{F}}^2 \ge \sigma_{\min}(\boldsymbol{U})^2 \|\boldsymbol{V}\|_{\mathrm{F}}^2 \ge c'^2 n(nr/2) \ge (c'^2/2) n^2 r$$

and

$$\|\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}\|_{\mathrm{F}}^2 \le \|\boldsymbol{U}\|_2^2 \|\boldsymbol{V}\|_{\mathrm{F}}^2 \le 8n^2 r.$$ □

Hence $\|(\alpha/\sqrt{n})\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2 = \Theta(nr)$ with a large probability. Therefore, we obtain that

$$\begin{aligned}&\mathbf{Pr}_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{G},\boldsymbol{\sigma}}[\|\mathcal{A}'((\alpha/\sqrt{n})\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}+\boldsymbol{G},\boldsymbol{\sigma})-(\alpha/\sqrt{n})\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \le c\|(\alpha/\sqrt{n})\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2]\\ &\quad\ge(1-\exp(-\Theta(n)))p.\end{aligned}$$

Thus there is some σ such that

$$\begin{aligned}&\mathbf{Pr}_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{G}}[\|\mathcal{A}'((\alpha/\sqrt{n})\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}+\boldsymbol{G},\sigma)-(\alpha/\sqrt{n})\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \le c\|(\alpha/\sqrt{n})\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2]\\ &\quad\ge p(1-\exp(-\Theta(n)))\end{aligned}$$

which implies that with probability $\ge p(1-\exp(-\Theta(n)))$,

$$\|(\sqrt{n}/\alpha)\mathcal{A}'((\alpha/\sqrt{n})\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}+\boldsymbol{G},\sigma)-\sum_{i=1}^{r}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}\|_{\mathrm{F}}^2 \le 2cn^2r.$$

By picking c small enough, we obtain using Theorem 5.3.4 that $t = \Omega(\log(n^2/k)/\log\log n)$ for algorithms with success probability $p \ge 9/10$. □

5.5 Lower Bounds for Other Problems

5.5.1 Spectral Low Rank Approximation

Theorem 5.5.1. *Given $n, r \in \mathbb{Z}$, if a t-round adaptive algorithm that performs $k \ge nr$ general linear measurements in each round is such that for every $n \times n$ matrix A, the algorithm outputs a rank r matrix B such that with probability $\ge 99/100$, $\|A-B\|_2 \le 2\sigma_{r+1}(A)$, then $t \ge c\frac{\log(n^2/k)}{\log\log(n)}$.*

We prove the following helper lemma that we use in our proof.

Lemma 5.5.2. *If A is a rank-r matrix and B is an arbitrary matrix satisfying $\|A-B\|_2 \le t$, then $\|A-[B]_r\|_2 \le 2t$, where $[B]_r$ denotes the best rank-r approximation of the matrix B in operator norm.*

Proof. As A has rank at most r, we have $t \ge \|A-B\|_2 \ge \|B-[B]_r\|_2$. Thus, $\|A-[B]_r\|_2 \le \|A-B\|_2 + \|B-[B]_r\|_2 \le 2t$. □

Proof of Theorem 5.5.1. By a standard reduction, it suffices to show that there is a distribution on $n \times n$ matrices for which any t-round *deterministic* algorithm which makes k general linear measurements in each round and outputs a 2-approximate spectral rank approximation satisfies $t \ge c\log(n^2/k)/\log\log(n)$.

Consider the $n \times n$ random matrix $\boldsymbol{M} = \boldsymbol{G} + s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ for $s = 500/\sqrt{n}$. Suppose there is a deterministic t-round algorithm Alg that makes k linear measurements and outputs rank-r matrix $K = \mathsf{Alg}(\boldsymbol{M})$ such that with probability $\geq 99/100$,

$$\|\boldsymbol{M} - K\|_2 \leq 2\sigma_{r+1}(\boldsymbol{M}) \leq 2\|\boldsymbol{G}\|_2.$$

This implies that $\|s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - K\|_2 \leq 3\|\boldsymbol{G}\|_2$ and $\|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - K/s\|_2 \leq 3\|\boldsymbol{G}\|_2/s$.

With probability $\geq 1 - \exp(-\Theta(n))$, we have $2\sqrt{n} \geq \|\boldsymbol{u}\|_2, \|\boldsymbol{v}\|_2 \geq \sqrt{n}/2$ and $\|\boldsymbol{G}\|_2 \leq 3\sqrt{n}$ simultaneously. By a union bound, we have that with probability ≥ 0.98, $\|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - K/s\|_2 \leq (9/500)n$. By Lemma 5.5.1, we have $\|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - [K/s]_1\|_2 \leq (9/250)n$ which implies that $\|\boldsymbol{u} \otimes \boldsymbol{v} - \mathrm{vec}([K/s]_1)\|_2^2 = \|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - [K/s]_1\|_{\mathrm{F}}^2 \leq 2\|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - [K/s]_1\|_2^2 \leq 2(9/250)^2 n^2$ where we used the fact that the matrix $\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - [K/s]_1$ has rank at most 2. Let $q \in \mathbb{R}^{n^2} = \mathrm{vec}([K/s]_1)/\|[K/s]_1\|_{\mathrm{F}}$ be a unit vector. We can show that

$$\langle q, \boldsymbol{u} \otimes \boldsymbol{v}\rangle^2 \geq n^2/64$$

by a simple application of the triangle inequality. Thus, using Alg we can construct a deterministic $t + 1$ round algorithm such that the squared projection of $\boldsymbol{u} \otimes \boldsymbol{v}$ onto the query space is at least $\geq n^2/64$ with probability $\geq 98/100$. By (5.3), we have that $t + 1 \geq c' \log(n^2/k)/\log\log(n)$, and therefore we have $t \geq c \log(n^2/k)/\log\log(n)$ for a small enough constant c. □

The following theorem states our lower bound for algorithms which output a spectral LRA for each matrix with high probability.

Theorem 5.5.3. *Given $n, r \in \mathbb{Z}$, if a t-round adaptive algorithm that performs $k \geq nr$ general linear measurements in each round is such that for every $n \times n$ matrix A, the algorithm outputs a rank r matrix B such that with probability $\geq 1 - 1/\mathrm{poly}(n)$, $\|A - B\|_2 \leq 2\sigma_{r+1}(A)$, then $t \geq c\log(n^2/k)$.*

Proof of Theorem 5.5.3. The proof of this lemma is similar to that of Theorem 5.5.1. The existence of a randomized t round algorithm that outputs a 2-approximate spectral norm LRA for every instance with probability $\geq 1 - 1/\mathrm{poly}(n)$ implies the existence of a deterministic algorithm that outputs a 2-approximate spectral norm LRA with probability $\geq 1 - 1/\mathrm{poly}(n)$ over the distribution of $\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$. As in the proof of the Theorem 5.5.1, this algorithm can be used to construct a $t + 1$ round algorithm that with probability $\geq 1 - 1/\mathrm{poly}(n)$ over the matrix $\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$, computes a unit vector q satisfying

$$\langle q, \boldsymbol{u} \otimes \boldsymbol{v}\rangle^2 \geq n^2/64.$$

Now, (5.4) implies that $t + 1 \geq c\log(n^2/k)$ for a small enough constant c. □

For the random matrix $\boldsymbol{M} = \boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ for a large enough constant α considered in the above theorem, we have that $(\sigma_1(\boldsymbol{M})/\sigma_2(\boldsymbol{M})) \geq 2$ with high probability. Now consider the random matrix

$$\boldsymbol{M}' = \begin{bmatrix} \boldsymbol{M} & 0 \\ 0 & 3\sqrt{n}\alpha I_{r-1} \end{bmatrix}.$$

With high probability, we have $\sigma_r(\boldsymbol{M}') = \sigma_1(\boldsymbol{M}) \geq (\alpha/2)\sqrt{n}$ and $\sigma_{r+1}(\boldsymbol{M}') = \sigma_2(\boldsymbol{M}) \leq 2\sqrt{n}$ implying $\sigma_{r+1}(\boldsymbol{M}')/\sigma_r(\boldsymbol{M}') \leq 1/2$. A proof similar to that of the above theorem now shows that algorithms using $k \geq nr$ general linear measurements in each round and outputting a 2-approximate rank-r LRA with probability $\geq 1 - 1/\text{poly}(n)$ for matrices M with $\sigma_{r+1}(M)/\sigma_r(M) \leq 1/2$ have a lower bound of $\Omega(\log(n^2/k))$ rounds. Moreover for $k \geq Cnr$ for a large enough constant C and $r = O(1)$, the randomized subspace iteration algorithm starting with a subspace of k/n-dimensions, after $O(\log(n^2/k))$ rounds outputs, with probability $\geq 1 - 1/\text{poly}(n)$, a 2-approximate rank-$r$ spectral norm LRA for all matrices satisfying $\sigma_{r+1}(M)/\sigma_r(M) \leq 1/2$. See [Gu15, Theorem 5.8] for a proof.

Note that the subspace iteration algorithm starting with a subspace of k/n-dimensions performs $(k/n) \cdot n = k$ general linear measurements in each round. Thus for $r = O(1)$ and $k \geq Cnr$ for a large enough constant C, the lower bound of $\Omega(\log(n^2/k))$ rounds for high probability spectral norm LRA algorithms for "well-conditioned" ($\sigma_{r+1}/\sigma_r \leq 1/2$) instances is tight up to constant factors and shows that general linear measurements offer no improvement over matrix-vector products for well-conditioned problems. Thus, we have the following theorem.

Theorem 5.5.4. *Any randomized t-round algorithm that with probability $\geq 1 - 1/\text{poly}(n)$ outputs a 2-approximate rank-r spectral norm LRA for any arbitrary matrix A satisfying $\sigma_r(A)/\sigma_{r+1}(A) \geq 2$, must have $t = \Omega(\log(n^2/k))$, where $k \geq nr$ is the number of linear measurements the algorithm makes in each round.*

Moreover, for $r = O(1)$, the subspace iteration algorithm [Gu15] matches the lower bound up to constant factors and outputs a 2-approximate spectral norm LRA for all such instances with probability $\geq 1 - 1/\text{poly}(n)$ in $O(\log(n^2/k))$ rounds.

5.5.2 Symmetric Spectral Norm Low Rank Approximation

An interesting property of the hard distributions from previous works [SEAR18, BHSW20] is that those distributions are supported on symmetric matrices, and they hence obtain lower bounds for algorithms that work even only on symmetric instances. Although our hard distribution is non-symmetric, we can construct a distribution supported only on symmetric matrices and show lower bounds on algorithms using generalized linear queries for symmetric matrices.

Theorem 5.5.5. *Given $n, r \in \mathbb{Z}$, if a t-round adaptive algorithm that performs $k \geq nr$ general linear measurements in each round is such that for every $n \times n$ matrix A, the algorithm outputs a rank $r \geq 2$ matrix B such that with probability $\geq 99/100$, $\|A - B\|_2 \leq 2\sigma_{r+1}(A)$, then $t \geq c\frac{\log(n^2/k)}{\log\log n}$.*

Proof. Consider the $2n \times 2n$ random matrix $\boldsymbol{M}'$ defined as

$$\boldsymbol{M}' = \begin{bmatrix} 0_{n\times n} & \boldsymbol{M} \\ \boldsymbol{M}^{\mathrm{T}} & 0_{n\times n} \end{bmatrix}$$

where $\boldsymbol{M} = \boldsymbol{G} + s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ for $s = 500/\sqrt{n}$ and all coordinates of $\boldsymbol{G}, \boldsymbol{u}, \boldsymbol{v}$ are independent standard Gaussian random variables. We have that for $i \in [n]$, $\sigma_{2i-1}(\boldsymbol{M}') = \sigma_{2i}(\boldsymbol{M}') = \sigma_i(\boldsymbol{M})$ and that any arbitrary k generalized linear measurements of the matrix $\boldsymbol{M}'$ can be simulated using $2k$ general linear measurements of $\boldsymbol{M}$. Now suppose an algorithm that uses k general linear measurements in each round outputs a rank $r \geq 2$ matrix K satisfying

$$\|\boldsymbol{M}' - K\|_2 \leq 2\sigma_{r+1}(\boldsymbol{M}') \leq 2\sigma_2(\boldsymbol{M}).$$

Let the matrix K be of the form

$$K = \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix}$$

As any sub-matrix of K has rank at most that of K, we obtain that K_2 is a rank-r matrix satisfying

$$\|\boldsymbol{M} - K_2\|_2 \leq 2\sigma_2(\boldsymbol{M}).$$

Thus, the existence of a t-round deterministic algorithm that uses k generalized queries in each round and outputs a constant factor approximation of rank r, spectral norm LRA, for the random matrix $\boldsymbol{M}'$ with probability $\geq 99/100$ implies the existence of a t-round algorithm that uses $2k$ generalized queries in each round and outputs a constant factor approximation of rank-r spectral norm LRA for the random matrix $\boldsymbol{M}$ with probability $\geq 99/100$. Now, as in the proof of Theorem 5.5.1, we obtain that

$$t \geq c\frac{\log(n^2/2k)}{\log\log n} \geq c'\frac{\log(n^2/k)}{\log\log n}.$$

We obtain the proof by appropriately scaling n in the statement. $\square$

The above theorem proves lower bounds for algorithms that solve constant factor rank-r spectral norm Low Rank Approximation (LRA) for all $r \geq 2$ even for symmetric instances. This leaves open just a lower bound on algorithms solving rank 1 spectral norm LRA for symmetric instances.

5.5.3 Schatten Norm Low Rank Approximation

We first note the following lemma which bounds the Schatten-p norm of an $n \times n$ Gaussian matrix.

Lemma 5.5.6 (Equation 3.3 in [LNW14])**.** *If $\boldsymbol{G}$ is an $n \times n$ matrix with independent entries sampled from $N(0,1)$ and $p \geq 2$, then with probability $\geq 9/10$, $\|\boldsymbol{G}\|_{\mathrm{S}_p} \leq 30n^{1/2+1/p}$.*

We now state the theorem that shows lower bounds for Schatten norm low rank approximation.

Theorem 5.5.7. *Given $n, r \in \mathbb{Z}$, if a t-round adaptive algorithm that performs $k \geq nr$ general linear measurements in each round is such that for every $n \times n$ matrix A, the algorithm outputs a rank r matrix B such that with probability $\geq 99/100$ we have*

$$\|A - B\|_{\mathrm{S}_p} \leq 2 \min_{\text{rank-}r\ X} \|A - X\|_{\mathrm{S}_p},$$

then

$$t \geq c\frac{\log(n^2/k)}{1 + (1/p)\log(n) + \log\log(n)}.$$

Proof. The proof is very similar to the proof of Theorem 5.5.1. Let $\boldsymbol{M} = \boldsymbol{G} + s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ for $s = \alpha/\sqrt{n}$ for α to be chosen later. Let Alg be a t-round deterministic algorithm that outputs a 2-approximate Schatten p-norm low rank approximation for $\boldsymbol{M}$ with probability $\geq 99/100$ over $\boldsymbol{M}$. If B is the matrix output by a 2-approximate Schatten-p LRA algorithm, then we have

$$\|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - B/s\| \leq \frac{2\|\boldsymbol{G}\|_{\mathrm{S}_p}}{s}$$

as $\min_{\text{rank-}r\ X} \|\boldsymbol{G} + s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - X\|_{\mathrm{S}_p} \leq \|\boldsymbol{G}\|_{\mathrm{S}_p}$. Using Lemma 5.5.6, with probability $\geq 9/10$ over $\boldsymbol{M}$, we have that

$$\|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - B/s\|_{\mathrm{S}_p} \leq 90n^{1+1/p}/\alpha.$$

By picking $\alpha = Bn^{1/p}$ for a large enough constant B, we obtain that

$$\|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - B/s\|_2 \leq \|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - B/s\|_{\mathrm{S}_p} \leq (9/250)n.$$

Similar to the proof of Theorem 5.5.1, we can construct a unit vector $q \in \mathbb{R}^{n^2}$ such that with probability $\geq 8/10$ over $\boldsymbol{M}$, we have

$$\langle q, \boldsymbol{u} \otimes \boldsymbol{v}\rangle^2 \geq n^2/64.$$

Using (5.3), we obtain that

$$t + 1 \geq c\frac{\log(n^2/k)}{1 + \log(\alpha) + \log\log(n)} \geq c\frac{\log(n^2/k)}{1 + (1/p)\log(n) + \log\log(n)}. \qquad \square$$

In particular, for $p = O(\log(n)/\log\log(n))$, this gives a lower bound of $\Omega(p(2-\log(k)/\log(n)))$ on the number of adaptive rounds required if an algorithm can query k general linear measurements in each round. Recently, Bakshi, Clarkson and Woodruff [BCW22, Algorithm 2] gave an algorithm for Schatten-p low rank approximation for arbitrary matrices that runs in $O(p^{1/2}\log(n))$[4] iterations to output a constant factor approximation. For instances with $\sigma_r(A)/\sigma_{r+1}(A) = 1 + \Omega(1)$, Saibaba [Sai19, Theorem 8] showed that randomized subspace iteration gives a constant factor approximation to rank-r Schatten-p norm LRA in $O(\log(n))$ iterations using nr linear measurements in each round.

5.5.4 Ky-Fan Norm Low Rank Approximation

Theorem 5.5.8. *Given $n, r \in \mathbb{Z}$, if a t-round adaptive algorithm that performs $k \geq nr$ general linear measurements in each round is such that, for every $n \times n$ matrix A, the algorithm outputs a rank-r matrix B such that with probability $\geq 99/100$ we have*

$$\|A - B\|_{\mathrm{F}_p} \leq 2 \min_{\text{rank-}r\ X} \|A - X\|_{\mathrm{F}_p},$$

then

$$t \geq c \cdot \log(n^2/k)/(\log(p) + \log\log(n)).$$

Proof. Again consider the same instance $\boldsymbol{M} = \boldsymbol{G} + s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ with $s = \alpha/\sqrt{n}$ for α chosen later. If K is a 2-approximate rank-r low rank approximation of $\boldsymbol{M}$ in Ky-Fan p norm, then

$$\|\boldsymbol{G} + s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - K\|_{\mathrm{F}_p} \leq 2\|\boldsymbol{G}\|_{\mathrm{F}_p}$$

which by the triangle inequality implies that $\|s\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - K\|_{\mathrm{F}_p} \leq 3\|\boldsymbol{G}\|_{\mathrm{F}_p}$. As $\|\boldsymbol{G}\|_2 \leq 2\sqrt{n}$ with high probability, we have that $\|\boldsymbol{G}\|_{\mathrm{F}_p} \leq 2p\sqrt{n}$ with high probability. Thus,

$$\|\boldsymbol{s}\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - K/s\|_2 \leq \|\boldsymbol{s}\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - K/s\|_{\mathrm{F}_p} \leq \frac{6pn}{\alpha}.$$

Picking $\alpha = Bp$ for a large enough constant B, we obtain that using the matrix K, we can construct

[4]Although their algorithm is stated to use $rp^{1/6}\log^2(n)$ matrix-vector products, this does not appear to have been optimized, and looking at the analysis it runs in at most $p^{1/2}\log(n)$ iterations, where each round queries at most r matrix-vector products.

a unit vector q such that $\langle q, \boldsymbol{u} \otimes \boldsymbol{v} \rangle^2 \geq n^2/64$, thus showing a

$$t = c \cdot \frac{\log(n^2/k)}{/}(\log(p) + \log\log(n))$$

round lower bound on any algorithm that performs k general linear measurements to output a 2-approximate rank-r approximation in Ky-Fan p norm. □

For $p = O(1)$, the Block Krylov iteration algorithm [MM15] gives an $O(1)$ approximate solution to the rank-r Ky-Fan norm low rank approximation problem in $O(\log(n))$ rounds while querying nr general linear measurements in each round. Thus, our lower bound on constant approximate algorithms for Ky-Fan norm LRA is optimal up to a $\log\log(n)$ factor for $r, p = O(1)$.

5.5.5 Frobenius Norm Low Rank Approximation

Theorem 5.5.9. *Given an $n \times n$ matrix A with $\sigma_1(A)/\sigma_2(A) \geq 2$, if a t-round algorithm outputs a rank-1 matrix B satisfying $\|A - B\|_{\mathrm{F}}^2 \leq (1 + 1/n)\|A - [A]_1\|_{\mathrm{F}}^2$ with probability $\geq 99/100$, then we have $t \geq c\log(n^2/k)/\log\log(n)$.*

First, we prove the following helper lemma.

Lemma 5.5.10. *Given $A \in \mathbb{R}^{n \times n}$, if a rank r matrix B is a $1 + 1/(n - r)$ approximate rank r Frobenius norm LRA, then*

$$\|A - B\|_2^2 \leq 2\sigma_{r+1}(A)^2.$$

Proof. We use the fact [Gu15, Theoem 3.4] that if a rank r matrix satisfies $\|A-B\|_{\mathrm{F}}^2 \leq \|A-[A]_r\|_{\mathrm{F}}^2 + \eta$, then $\|A - B\|_2^2 \leq \|A - [A]_r\|_2^2 + \eta$. If B is a $1 + 1/(n - r)$ approximate solution for rank r Frobenius norm LRA, we have

$$\begin{aligned}
\|A - B\|_{\mathrm{F}}^2 &\leq \left(1 + \frac{1}{n - r}\right)\|A - [A]_r\|_{\mathrm{F}}^2 \\
&\leq \|A - [A]_r\|_{\mathrm{F}}^2 + \frac{\sigma_{r+1}(A)^2 + \ldots + \sigma_n(A)^2}{n - r} \\
&\leq \|A - [A]_r\|_{\mathrm{F}}^2 + \sigma_{r+1}(A)^2
\end{aligned}$$

which implies $\|A - B\|_2^2 \leq \|A - [A]_r\|_2^2 + \sigma_{r+1}(A)^2 = 2\sigma_{r+1}(A)^2$. □

Proof of Theorem 5.5.9. Using the above lemma, we have that whenever $\|A - B\|_{\mathrm{F}}^2 \leq (1 + 1/n)\|A - [A]_1\|_{\mathrm{F}}^2$ for a rank 1 matrix B, then $\|A - B\|_2^2 \leq 2\|A - B\|_2^2$. Consider the random matrix $\boldsymbol{M} = \boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ considered in Theorem 5.5.1. With probability $\geq 99/100$, we have that $\sigma_1(\boldsymbol{M})/\sigma_2(\boldsymbol{M}) \geq 2$ by picking α to be a large enough constant.

A t-round randomized algorithm for $(1 + 1/n)$-approximate rank-1 Frobenius norm LRA thus implies the existence of a $(t + 1)$ round deterministic algorithm that with probability $\geq 98/100$ over the random matrix $\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ makes general queries such that the squared projection of $\boldsymbol{u} \otimes \boldsymbol{v}$ onto the query space of the algorithm exceeds $n^2/64$ using similar arguments as in proof of Theorem 5.5.1. Now, using (5.3), we obtain that

$$t \geq c \log(n^2/k)/\log\log(n)$$

for a small enough constant c. □

For matrices A with $\sigma_1(A)/\sigma_2(A) \geq 2$, subspace iteration algorithm gives a $1+1/n$-approximate solution in $O(\log(n))$ rounds using n-linear measurements [GLO81, MM15]. The above lower bound shows that if an algorithm is allowed $o(\log(n)/\log\log(n))$ adaptive rounds, then the algorithm has to make $n^{2-o(1)}$ linear measurements in each round, which is almost as many measurements as is required to read the entire matrix.

5.5.6 Lower Bound for Reduced-Rank Regression in Spectral Norm

Given matrices A, B, and a rank parameter r, the reduced-rank regression problem in spectral norm is defined as

$$\min_{\text{rank-}r\ X} \|AX - B\|_2.$$

Taking $A = I$, we have that the spectral norm low rank approximation is a special case of the reduced-rank regression problem. Thus we have the following lower bound on the number of rounds of an adaptive algorithm that outputs a 2-approximate solution for the reduced-rank regression problem.

Theorem 5.5.11. *If a t-round adaptive algorithm which given arbitrary $n \times n$ matrices A, B and a parameter r outputs a 2-approximate solution for the reduced-rank regression problem with probability $\geq 9/10$, then $t = \Omega(\log(n^2/k)/\log\log(n))$ where k is the number of linear measurements of the matrices A and B that the adaptive algorithm performs in each of the t rounds.*

With nr adaptive linear measurements in each round, the above theorem gives a lower bound of $\Omega(\log(n/r)/\log\log(n))$ on the number of rounds required to obtain a factor 2 approximation. In Chapter 6, we obtain an algorithm for reduced-rank regression that outputs constant factor approximations to the reduced-rank problem in $O(\text{poly}(\log n))$ (ignoring polylogarithmic factors from condition numbers) adaptive rounds using nr linear measurements in each round.

If the matrix $B = M + P$ has the structure of a planted matrix studied in this chapter with P being a rank r matrix and $\|P\|_2 > 10\|B - [B]_r\|_2$ and if the matrix B is well conditioned, then we can find an $O(1)$ approximate solution to reduced-rank regression problem in $O(\log(n) + \log(\|P\|_2/\mathrm{OPT}))$ rounds, where OPT is the optimal value of the reduced-rank regression problem.

First, we find a rank r matrix P' such that $\|B-P'\|_2 \le 2\|B-[B]_r\|_2$ in $O(\log(n))$ adaptive rounds using the Block Krylov iteration algorithm of [MM15], which queries nr linear measurements in each round. Now, for any matrix X,

$$\|AX - P'\|_2 = \|AX - B\|_2 \pm \|B - P'\|_2 = \|AX - B\|_2 \pm 2 \cdot \mathsf{OPT}.$$

Let $P' = U\Sigma V^{\mathrm{T}}$ be the singular value decomposition with matrix $U\Sigma$ having r columns. Then using high precision regression routines (see [Woo14] and references therein), we find a matrix $\widetilde{X}$ such that

$$\|A\widetilde{X} - AA^{+}U\Sigma\|_{\mathrm{F}}^2 \le \varepsilon\|U\Sigma\|_{\mathrm{F}}^2 \le 10r\varepsilon\|P\|_2^2.$$

in $O(\log(1/\varepsilon))$ iterations, where each iteration queries nr linear measurements. Again, using the triangle inequality, we have

$$\|A\widetilde{X}V^{\mathrm{T}} - P'\|_2 = \|A\widetilde{X} - U\Sigma\|_2 \le \|AA^{+}U\Sigma - A\widetilde{X}\|_2 + \|AA^{+}U\Sigma - U\Sigma\|_2.$$

If X^* is the optimal solution for the reduced rank regression problem, we have

$$\begin{aligned}\|AA^{+}U\Sigma - U\Sigma\|_2 &= \|AA^{+}U\Sigma V^{\mathrm{T}} - U\Sigma V^{\mathrm{T}}\|_2\\ &\le \|AX^* - P'\|_2\\ &\le \|AX^* - B\|_2 + 2\cdot\mathsf{OPT} = 3\cdot\mathsf{OPT}.\end{aligned}$$

We also have $\|AA^{+}U\Sigma - A\widetilde{X}\|_2 \le \|AA^{+}U\Sigma - A\widetilde{X}\|_{\mathrm{F}} \le O(\sqrt{r\varepsilon}\|P\|_2)$. We set $\varepsilon = (\mathsf{OPT}/\|P\|_2)^2/Kr$ for a large enough constant K and obtain that $\|AA^{+}U\Sigma - A\widetilde{X}\|_2 \le \mathsf{OPT}$. Overall, we have $\|A\widetilde{X}V^{\mathrm{T}} - B\|_2 \le \|A\widetilde{X}V^{\mathrm{T}} - P'\|_2 + 2\cdot\mathsf{OPT} \le 6\cdot\mathsf{OPT}$. Thus, we obtain a 6-approximate solution in $O(\log(n) + \log(\|P\|_2/\mathsf{OPT}))$ iterations given that A is well-conditioned and B is of the form $M+P$ with $\|P\|_2 \ge 10\|M\|_2$ for a rank r matrix P. Thus the lower bound is near optimal for algorithms that output $O(1)$ approximate solution for planted models and well-conditioned coefficient matrices.

5.5.7 Lower Bound for Approximating the i-th Singular Vectors

Theorem 5.5.12. *If a t-round algorithm, given an $n\times n$ matrix A and $i \le n/2$, outputs unit vectors u_i' and v_i' such that with probability $\ge 9/10$,*

$$\|u_i' - u_i\|_2 \le 1/10 \text{ and } \|v_i' - v_i\|_2 \le 1/10$$

where u_i and v_i are respectively the i-th left and right singular vectors of the matrix A, then

$$t = \Omega(\log(n^2/k)/\log\log(n)),$$

where k is the number of linear measurements the algorithm performs on the matrix A in each of the t rounds.

We also have a $t = \Omega(\log(n^2/k))$ lower bound on the number of rounds required for an adaptive algorithm to compute approximations to the i-th left and right singular vectors with probability $\geq 1 - 1/\mathrm{poly}(n)$.

Proof. Consider the $n \times n$ random matrix $\boldsymbol{M} = \boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ for a large enough constant α. The existence of a t-round algorithm as in the statement implies that there is a deterministic t-round algorithm that outputs approximations to singular vectors of the matrix $\boldsymbol{M}$ with probability $\geq 9/10$. We then have that $10\|\boldsymbol{G}\|_2 \leq \|(\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\|_2$ and $\|\boldsymbol{u}\|_2, \|\boldsymbol{v}\|_2 \geq \sqrt{n}/2$ with large probability. Condition on this event. Let $v_1 \in \mathbb{R}^n$ be the top right singular vector of the matrix $\boldsymbol{M}$. We have

$$(9/10)\|(\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\|_2 \leq \|(\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\|_2 - \|\boldsymbol{G}\|_2 \leq \|\boldsymbol{M}\|_2 = \|(\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}})v_1\|_2.$$

By the triangle inequality, we have

$$\begin{aligned}\|(\alpha/\sqrt{n})\boldsymbol{u}(\boldsymbol{v}^{\mathrm{T}}v_1)\|_2 &\geq \|(\boldsymbol{G} + (\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}})\|_2 - \|\boldsymbol{G}v_1\|_2\\ &\geq (9/10)\|(\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\|_2 - (1/10)\|(\alpha/\sqrt{n})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\|_2\\ &= (8/10)\cdot(\alpha/\sqrt{n})\|\boldsymbol{u}\|_2\|\boldsymbol{v}^{\mathrm{T}}\|_2.\end{aligned}$$

Thus, we obtain that $|\boldsymbol{v}^{\mathrm{T}}v_1| \geq 4/5\|\boldsymbol{v}\|_2$. Similarly, if u_1 is the top left singular vector, we have that $|\boldsymbol{u}^{\mathrm{T}}u_1| \geq 4/5\|\boldsymbol{u}\|_2$. Suppose v_1' is a unit vector such that $\|v_1 - v_1'\|_2 \leq 1/10$. Then, $|\boldsymbol{v}^{\mathrm{T}}v_1'| \geq |\boldsymbol{v}^{\mathrm{T}}v_1| - (1/10)\|\boldsymbol{v}\|_2 \geq (7/10)\|\boldsymbol{v}\|_2$. Similarly, $|\boldsymbol{u}^{\mathrm{T}}u_1'| \geq (7/10)$ which implies

$$\langle u_1' \otimes v_1', \boldsymbol{u} \otimes \boldsymbol{v}\rangle^2 = \langle u_1', \boldsymbol{u}\rangle^2\langle v_1', \boldsymbol{v}\rangle^2 \geq (49/100)\|\boldsymbol{u}\|_2^2\|\boldsymbol{v}\|_2^2 \geq n^2/10.$$

Thus, with a large probability over the random matrix $\boldsymbol{M}$, approximations to the top left and right singular vectors of the matrix $\boldsymbol{M}$ can be used to construct an n^2-dimensional unit vector q such that $\langle q, \boldsymbol{u} \otimes \boldsymbol{v}\rangle^2 \geq n^2/10$. Thus, we have a $t+1$ round deterministic algorithm such that the squared projection of $\boldsymbol{u} \otimes \boldsymbol{v}$ onto the query space of the algorithm is $\geq n^2/10$ with probability $\geq 8/10$ over the random matrix $\boldsymbol{M}$. Now, (5.3) implies that $t + 1 \geq c\log(n^2/k)/\log\log(n)$ for a small enough constant c.

To extend the above lower bound to approximating i-th singular vector for all $i \leq n/2$, we can use the following instance:

$$\boldsymbol{M}' = \begin{bmatrix}\boldsymbol{M} & 0\\ 0 & S\end{bmatrix}$$

where $\boldsymbol{M}$ is the $(n/2) \times (n/2)$ random matrix $\boldsymbol{G} + (\alpha/\sqrt{n/2})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ and S is a deterministic diagonal matrix chosen such that the top singular vectors of $\boldsymbol{M}$ correspond to i-th singular vectors of the matrix $\boldsymbol{M}'$. If S is chosen to be the diagonal matrix with $i - 1$ values equal to $10\sqrt{n}\alpha$ and the rest of the entries are set to 0, we have that with high probability that the i-th left and right singular vectors of $\boldsymbol{M}'$ correspond to the top left and right singular vectors of $\boldsymbol{M}$ as $\|M\|_2 \le 10\sqrt{n}\alpha$ with high probability. Now the existence of a t-round randomized algorithm to approximate i-th left and right singular vectors of an $n \times n$ matrix for $i \le n/2$ with probability $\ge 9/10$ implies that there is a deterministic t-round algorithm that approximates the top left and right singular vectors of the $(n/2) \times (n/2)$ matrix $\boldsymbol{M}$ with probability $\ge 8/10$. From the above argument we have $t + 1 = \Omega(\log((n/2)^2/k)/\log\log(n/2)) = \Omega(\log(n^2/k)/\log\log(n))$.

The lower bounds for algorithms that output approximations to singular vectors with probability $\ge 1 - 1/\text{poly}(n)$ follow similarly using the high probability lower bounds from (5.4). □

5.5.8 Lower Bounds for Sparse Matrices

The following theorem gives a lower bound of $\Omega(\log(m/k))$ rounds on adaptive algorithms that compute 2-approximate spectral norm LRA for matrices with at most m nonzero entries.

Theorem 5.5.13. *Any t round randomized algorithm that computes, given an arbitrary $n \times n$ matrix A with at most m nonzero entries, a 2-approximate rank r spectral norm LRA of A with probability $\ge 1 - 1/\text{poly}(m)$, must have $t = \Omega(\log(m/k))$, where k is the number of general linear measurements queried by the algorithm in each of the t rounds.*

Proof. We consider the distribution of the $\sqrt{m} \times \sqrt{m}$ random matrix $\boldsymbol{G} + (\alpha/m^{1/4})\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ for a large enough constant α and embed this instance into an $n \times n$ matrix. Clearly, all the matrices drawn from this distribution have at most m nonzero entries. And from Theorem 5.5.3, any deterministic algorithm using k linear measurements in each round and computing a 2-approximate spectral norm approximation for a matrix drawn from this distribution, with probability $\ge 1 - 1/\text{poly}(m)$, must use $\Omega(\log((\sqrt{m})^2/k)) = \Omega(\log(m/k))$ rounds.

Thus by Yao's minimax lemma, any randomized algorithm that outputs a 2-approximate spectral norm approximation with probability $\ge 1 - 1/\text{poly}(m)$ for an arbitrary matrix with m nonzero entries must use $t = \Omega(\log(m/k))$ rounds. □

The above theorem implies that any algorithm with only $O(1)$ adaptive rounds must query $\Omega(m)$ linear measurements. This lower bound is tight up to constant factors as with $2m$ linear measurements, since all the m non-zero entries of any arbitrary matrix can be computed in just 1 round using a Vandermonde matrix [FR13, Theorem 2.14].

5.6 Conclusions and Open Questions

In this work, we study the measurements-vs-number of rounds trade off for estimating a low rank matrix planted in Gaussian noise in the linear measurements model and use the result to obtain lower bounds on the number of rounds necessary for computing a Spectral norm low rank approximation, Frobenius norm low rank approximation problem etc. Our lowerbounds show that if one uses

$n^{2-\Omega(1)}$ linear measurements in each round, then the algorithms have to run for $\widetilde{\Omega}(\log n)$ rounds thereby proving that linear measurements are not much more powerful than matrix-vector products for many linear algebra problems.

The block Krylov iteration algorithm of [MM15] uses k matrix-vector products in each round and a total of $O(\log n/\sqrt{\varepsilon})$ rounds to compute a $1+\varepsilon$ approximate rank-k Spectral norm low rank approximation for an $n \times n$ matrix. [BN23] show that for $k = 1$, the algorithm block Krylov iteration algorithm is optimal in terms of the number of rounds. A major open question is if it is optimal for all values of k. Another interesting question is if linear measurements let us obtain better dependence on ε.

Chapter 6

Reduced-Rank Regression with Operator Norm Error

6.1 Introduction

Given an $n \times c$ matrix A, an $n \times d$ matrix B, and an integer parameter k, the reduced-rank regression problem asks to find a rank-k matrix $X \in \mathbb{R}^{c \times d}$ for which $\|AX - B\|$, where $\| \cdot \|$ denotes some matrix norm. A standard motivation is that by constraining X to have rank at most k, the solution X can be represented using only $(c+d)k$ parameters rather than $c \cdot d$ parameters. Another important motivation is that the rank constraint provides regularization on the solution, which often leads to better generalization. Yet another motivation is that the solution X can be explained by at most k latent factors, and one can try to interpret the latent factors, plot them [BL94], and so on. This is commonly done in ecology, where reduced-rank regression is known as redundancy analysis [LA99], and is a type of ordination method [KBW⁺19]. For a survey, we refer the reader to the textbook by Velu and Reinsel [VR13] devoted to reduced-rank regression.

The $\min_{\text{rank-}k\ X} \|AX - B\|$ problem is only known to have a closed form solution when the error measure is the Frobenius norm. In this case, the solution is given by $X = A^{+}[AA^{+}B]_k$ (see, e.g., [FT07]). Here for a matrix M, recall that $[M]_k$ denotes the best rank k approximation for M in Frobenius norm and M^{+} denotes the Moore-Penrose pseudo-inverse. This has a natural geometric interpretation - project each of the columns of B onto the column span of A and find the best rank-k approximation to the projected matrix. By the Pythagorean theorem, one can show there is no loss in this approach, as the optimal cost decomposes into the sum of squared distances of columns of B to the column span of A followed by the best rank-k approximation to the projected matrix inside the column span of A.

In a number of applications, the Frobenius norm is not the right measure. For example, in cancer genetics more robust versions are desired, and versions based on the sum of Euclidean lengths instead of the sum of squared Euclidean lengths are sometimes used [SC17]. Still, in other applications, the operator norm error solution may give a solution of much better quality. Indeed, if B has a heavy tail of singular values, as is common for data analysis and learning applications, then it has no good rank-k approximation, much less one in the column span of A, and consequently, outputting an X' with $\|AX' - B\|_F^2 \le (1+\varepsilon)\|AX_F - B\|_F^2$, where X_F is the optimal Frobenius norm solution, may be meaningless as one could just set $X' = 0$. Indeed, this is sometimes a motivation (see, e.g., [MM15]) for the low rank approximation problem with operator norm error, which is a special case of our problem when $A = B$, and a number of works [HMT11, JLSW20, KL15, SKT14] suggest considering operator norm error in certain contexts.

It is tempting to think that the optimal Frobenius norm solution holds is also optimal for the reduced-rank regression problem for other unitarily invariant norms, such as the operator norm. However, one can show this is not the case. Indeed, let X_F be the solution to $\min_{\text{rank-}k\ X} \|AX - B\|_F$. It was shown by [Bou11] that this is a $\sqrt{2}$-approximation, namely, that $\|AX_F - B\|_2 \le \sqrt{2} \cdot \mathrm{OPT}$ where $\mathrm{OPT} = \min_{\text{rank-}k\ X} \|AX - B\|_2$. Unfortunately, the $\sqrt{2}$ factor is tight and there are instances where the Frobenius norm solution really does give at best a $\sqrt{2}$-approximation. Suppose, for example[1]

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ and } B = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1+\gamma \end{bmatrix}.$$

For the problem $\min_{\text{rank-}1\ X} \|AX - B\|_F$, the optimum solution is

$$X_F = \begin{bmatrix} 0 & 0 \\ 0 & 1+\gamma \end{bmatrix}, \text{ with } AX_F - B = -\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

and thus, $\|AX_F - B\|_2 = \sqrt{2}$. On the other hand, for

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \ AX - B = -\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1+\gamma \end{bmatrix},$$

and so $\|AX - B\|_2 = (1+\gamma)$. As $\gamma \to 0$, the approximation factor becomes arbitrarily close to $\sqrt{2}$.

We note that the reduced-rank regression problem in operator norm is non-convex in X due to the rank constraint, and it is not even clear this problem can be solved in polynomial time. Of the few techniques that are known for rank-constrained optimization, they do not apply here. One common method is alternating minimization, writing the problem above as $\min_{U,V} \|AUV - B\|_2$, where $U \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times d}$. The idea is to fix U, then solve for V, then fix V and solve for U, and repeat. When U is fixed, then $V = (AU)^+B$ is the optimum, and when V is fixed, the solution

[1]We thank Ankur Moitra for pointing out this example to us.

turns out to be $U = A^+ BV^+$, though this is not as obvious, see (1.3) in [Mah07], taking $p \to \infty$, for a proof. It turns out if one initializes with the Frobenius norm solution U, V, then each of these operations does not change U or V, and so by the example above, alternating minimization gives at best a $\sqrt{2}$-approximation. Other techniques include sketching to a small problem, and solving the small problem in the sketch space; sketches are well-known not to apply to operator norm low rank approximation problems, motivating the first open question in [Woo14].

This issue of polynomial time solvability was raised in the control theory literature by Sou and Rantzer [SR12], where a $(1+\varepsilon)$-approximation was obtained, but the time required to find the solution was at least the time to perform a singular value decomposition (SVD) on matrices A and B, which is prohibitive for large n, c, and d. This is a common setting of parameters and indeed, one of the motivations for constraining X to have rank at most k in the first place. This motivates the question:

"*Are there fast algorithms for reduced-rank regression with operator norm error?*"

6.1.1 Main Result

We answer the question above by designing a new randomized algorithm running in time

$$O\left(\left(\frac{\mathrm{nnz}(B)\cdot k}{\varepsilon} + \frac{\mathrm{nnz}(A)\cdot k}{\varepsilon^{1.5}} + \frac{c^2 k}{\varepsilon^{1.5}} + \frac{(n+d)k^2}{\varepsilon}\right)\cdot \mathrm{polylog}(\kappa(B), n, d, k, 1/\varepsilon) + c^\omega\right).$$

Here, $\kappa(B)$ denotes $\sigma_1(B)/\sigma_{k+1}(B)$. This significantly improves over Sou and Rantzer's polynomial time result, which takes $\Omega(nd^2 + nc^2)$ time.

We note that spectral low rank approximation is a special case in which $A = B$, and the best known upper bound is $O(\mathrm{nnz}(A)\cdot k/\sqrt{\varepsilon})$ for this problem, up to logarithmic factors [MM15]. A major open question in randomized numerical linear algebra is to improve this bound (see, e.g., Open Question 1 of [Woo14]), or show that it is not possible. We note that for $k = 1$, in the matrix-vector query model, $\Omega(1/\sqrt{\varepsilon})$ queries is known to be required [BN23]. Another important point is that when $n = c$ and $d = 1$, this is just the time to solve an arbitrary linear system, for which the best known time is c^ω. Improving either spectral low rank approximation or linear system solving is a major open question, and barring that, our algorithm is optimal up to a $1/\varepsilon$ factor and polylogarithmic factors involving matrix dimensions and condition numbers.

6.1.2 Our Techniques

Let $\mathrm{OPT} \coloneqq \inf_{\text{rank-}k\ X} \|AX - B\|_2$, β be such that $(1+\varepsilon)\mathrm{OPT} \le \beta \le (1+2\varepsilon)\mathrm{OPT}$, and let $\Delta \coloneqq B^{\mathrm{T}}(I - AA^+)B$. The work of Sou and Rantzer [SR12] shows that $X_\beta = A^+[AA^+B(\beta^2 I - \Delta)^{-1/2}]_k(\beta^2 I - \Delta)^{1/2}$ satisfies $\|AX - B\|_2 < \beta$. For completeness, we give a short proof of this fact. It is not a priori clear how to extract a fast algorithm from this expression. Multiplying out all the matrices, computing an inverse square root, and taking an SVD would take a prohibitive amount of time which is essentially the algorithm of [SR12].

We instead show that not only the best rank k approximation of the matrix $AA^+B(\beta^2 I - \Delta)^{-1/2}$, but even a $1+\varepsilon$ approximation in spectral norm yields an overall solution of cost at most $\beta(1+O(\varepsilon))$. To obtain such a $1+\varepsilon$ approximation, we next try to apply the iterative method of [MM15] which computes the Krylov matrix $K = [C \cdot G, (CC^{\mathrm{T}}) \cdot C \cdot G, (CC^{\mathrm{T}})^2 \cdot C \cdot G, \dots, (CC^{\mathrm{T}})^{(q-1)/2} \cdot C \cdot G]$ where G is a Gaussian matrix with k columns, $q = O(\log(d/\varepsilon)\sqrt{1/\varepsilon})$ is an odd integer, and $C = AA^+B(\beta^2 I - \Delta)^{-1/2}$. The first problem with this approach is that we have to compute the matrix product CG and to do this, in each iteration we need to (1) multiply by the square root of an inverse (multiplication by $(\beta^2 I - \Delta)^{-1/2}$), and then (2) project onto the column span of A (multiplication by AA^+).

Computing exact matrix-vector products with the matrices AA^+ and $(\beta^2 I - \Delta)^{-1/2}$, is slow when c, d are large, and finding the matrices AA^+ and $(\beta^2 I - \Delta)^{-1/2}$ takes at least $\Omega(nc^2 + \mathrm{nnz}(B) \cdot c + d^{\omega})$ time. To avoid such a running time, we show that the Block Krylov Iteration algorithm of Musco and Musco [MM15] works even with approximate matrix-vector products i.e., we only need algorithms to compute vectors $C \circ v$ and $C^{\mathrm{T}} \circ v'$ for arbitrary vectors v, v' such that $\|C \circ v - Cv\|_2$ and $\|C^{\mathrm{T}} \circ v' - C^{\mathrm{T}}v'\|_2$ are small. Here and in rest of the chapter, we use the notation $M \circ v$ (resp. $M^{\mathrm{T}} \circ v'$) to denote an approximation to the matrix-vector product Mv (resp. $M^{\mathrm{T}}v'$).

An important idea of Musco and Musco [MM15] is that the Krylov matrix K spans a rank k matrix $p(C)G = \sum_{\text{odd } i \le q} p_i (CC^{\mathrm{T}})^{(i-1)/2} G$, where p is a polynomial, such that projecting the columns of the matrix C onto the column span of $p(C)G$ gives a good rank k approximation for the matrix C. To prove that the algorithm works even with approximate matrix-vector products, we first show that the approximations computed to matrices $(CC^{\mathrm{T}})^{(i-1)/2}CG$ for $i = 1, \dots, q$ are good enough to imply that the approximate Krylov matrix K' spans a matrix Apx that is close to the matrix $p(C)G$ in Frobenius norm. To then conclude that the column space of Apx is also a good subspace to project the matrix C onto, we need to show that $(\mathrm{Apx})(\mathrm{Apx})^+ \approx (p(C)G)(p(C)G)^+$. We prove a simple lemma that shows if $\|p(C)G - \mathrm{Apx}\|_{\mathrm{F}}$ is small, and $p(C)G$ has a good condition number, then $\|(p(C)G)(p(C)G)^+ - (\mathrm{Apx})(\mathrm{Apx})^+\|_2$ is small. Crucially, as G is a Gaussian matrix that has, with good probability a good condition number, we only have to bound $\sigma_1(p(C))/\sigma_k(p(C))$ to obtain a bound on the condition number of $p(C)G$. Using several properties of Chebyshev polynomials used to define the polynomial $p(x)$, we show that $\sigma_1(p(C))/\sigma_k(p(C))$ can be bounded in terms of $\kappa = \sigma_1(C)/\sigma_{k+1}(C)$, which finally shows that the k-dimensional column span of Apx is also a good subspace to project the columns of C.

As the parameters of the polynomial $p(x)$ are unknown, we cannot actually compute the matrix Apx and then project C onto the column span. But using the fact that K' spans Apx, we can conclude, similarly to the arguments of [MM15], that the best rank k Frobenius norm approximation of C in the span of K' is a good rank k approximation to C. Using the oracle to compute approximate matrix-vector products with the matrix C, we recover a $1+\varepsilon$ approximation to the best rank k Frobenius norm approximation of C inside the span of K', which we then show is a $1+\varepsilon$ approximation to a spectral norm low rank approximation of matrix C. Our analysis that the Block Krylov Iteration algorithm works with approximate matrix-vector products could help justify why the Block Krylov Iteration algorithm works well when using finite precision arithmetic rather than exact arithmetic. Our results address the comments of [MMS18] about the stability of block Lanczos based methods for problems such as low rank approximation. Though several analyses of the noisy power method have been done previously [BDWY16, HP14, HR13], where each intermediate computation is corrupted by Gaussian noise, we are not aware of an analysis that works for worst case corruption. Also, previous work bounds the amount of Gaussian noise that can be added in terms of a gap between σ_k and σ_{k+1}, which can be 0, and would not work for our analysis.

We return to the task at hand, i.e., of computing a low rank approximation of $AA^+B(\beta^2I-\Delta)^{-1/2}$. We show that we can replace the matrix $(\beta^2I-\Delta)^{-1/2}$ with the matrix $(1/\beta)r(\Delta/\beta^2)$, where $r(x)$ is a polynomial of degree $\widetilde{O}(1/\sqrt{\varepsilon})$, using polynomial approximation techniques based on Chebyshev polynomials (see, e.g., [SV14] and the references therein). Here we crucially use the fact that $(1+2\varepsilon)\mathrm{OPT} \geq \beta \geq (1+\varepsilon)\mathrm{OPT} \geq (1+\varepsilon)\|(I-AA^+)B\|_2$ to lower bound the minimum singular value of the matrix $(\beta^2I-\Delta)$, thereby obtaining an upper bound on the number of terms required to approximate $(I-(\Delta/\beta^2))^{-1/2}$ with a Taylor series. Then we replace each monomial in the Taylor series with a low degree polynomial approximation to construct a polynomial $r(x)$. The replacement of $(\beta^2I-\Delta)^{-1/2}$ with the matrix $r(\Delta/\beta^2)$ is done as we can give very fast algorithms to approximately multiply a vector with the matrix $r(\Delta/\beta^2)$, as discussed below.

Let $\mathcal{M}' = AA^+B\cdot r(\Delta/\beta^2)$. Recall $\Delta = B^{\mathrm{T}}(I-AA^+)B$. To approximate the matrix-vector product Δu for an arbitrary vector u, we need only approximate $B^{\mathrm{T}}AA^+Bu$, since $B^{\mathrm{T}}Bu$ can be computed exactly in $\mathrm{nnz}(B)$ time. For computing an approximation to $AA^+(Bu)$, we use fast sketching-based preconditioning methods for linear regression, which show, given an arbitrary vector b and accuracy parameter $\varepsilon_{\mathrm{reg}}$, how to find a vector x for which $\|Ax-AA^+b\|_2 \leq \varepsilon_{\mathrm{reg}}\|(I-AA^+)b\|_2$ in time $O((\mathrm{nnz}(A)+c^2)\log(1/\varepsilon_{\mathrm{reg}})+c^{\omega})$, where $\omega \approx 2.376$ is the exponent of matrix multiplication [CW17, MM13, NN13]. We note that we only need to pay the c^{ω} time once to compute a preconditioner, after which each regression problem takes $O((\mathrm{nnz}(A)+c^2)\log(1/\varepsilon_{\mathrm{reg}}))$ time. This algorithm to approximately compute Δu for an arbitrary vector u is extended to approximate $r(\Delta/\beta^2)\cdot v$ for an arbitrary v. After approximating the product $r(\Delta/\beta^2)\cdot v$ with a vector y, we approximate the vector AA^+By again using the sketching-based preconditioning methods for linear regression.

Similarly, we also give an algorithm to approximate $\mathcal{M}'^{\mathrm{T}}v'$ for an arbitrary vector v'. Thus, as discussed above, we can obtain using a Block Krylov algorithm, a matrix Z with orthonormal columns for which $\|ZZ^{\mathrm{T}}\mathcal{M}'-\mathcal{M}'\|_2 \leq (1+\varepsilon)\sigma_{k+1}(\mathcal{M}')$ and then conclude that

$$\|AA^+Z(AA^+Z)^+B-B\|_2 \leq (1+O(\varepsilon))\beta = (1+O(\varepsilon))\mathrm{OPT}$$

and that the rank k matrix $X = A^+Z(AA^+Z)^+B$ is a $1+O(\varepsilon)$ approximation for the problem

$\min_{\text{rank-}k} \|AX - B\|_2$.

The time complexity of our algorithm depends logarithmically on $\kappa(B) = \sigma_1(B)/\sigma_{k+1}(B)$ and $\kappa(AA^+B) = \sigma_1(AA^+B)/\sigma_{k+1}(AA^+B)$. We show that if $\widetilde{B} = B + \alpha GF^{\mathrm{T}}$ where G is an $n \times (k+1)$ random Gaussian matrix and F^{T} has $k+1$ orthonormal rows, then for a suitable value of α, the condition number $\kappa(AA^+\widetilde{B}) \le (Cn/\varepsilon)\kappa(B)$ for a constant C. We also show that a $1+\varepsilon$ approximation for reduced rank regression computed using the matrix $\widetilde{B}$ is a $1 + O(\varepsilon)$ approximation for reduced rank regression on matrix B, thus removing the dependence on $\kappa(AA^+B)$. Note that matrix-vector products with $\widetilde{B}$ can be computed in $\mathrm{nnz}(B) + (n+d)k$ time.

Our final dependence on ε in the running time is $1/\varepsilon^{3/2}$, ignoring polylogarithmic factors, where a factor of $1/\sqrt{\varepsilon}$ is from the number of iterations in the Block Krylov Iteration algorithm of [MM15], a factor of $1/\sqrt{\varepsilon}$ is from the degree of the polynomial $r(x)$, which is used to approximate matrix $(\beta^2 I - \Delta)^{-1/2}$ with a matrix $r(\Delta/\beta^2)$, and a factor of $1/\sqrt{\varepsilon}$ is due to the running time of high-precision regression methods based on the accuracy with which the approximate matrix products need to be computed.

6.2 Notation and Preliminaries

For matrices M and M' of the same dimensions, $\langle M, M' \rangle$ denotes $\mathrm{tr}(M^{\mathrm{T}}M') = \sum_{i,j} M_{i,j}M'_{i,j}$. We use the following standard facts repeatedly: for any matrix M, (1) $\|M\|_2 \le \|M\|_{\mathrm{F}}$, (2) $\|M\|_{\mathrm{F}} \le \sqrt{\mathrm{rank}(M)}\|M\|_2$ and (3) $\mathbb{P}_M = MM^+$. For any matrices A, B and C, (i) $\mathrm{tr}(ABC) = \mathrm{tr}(BCA)$, (ii) $\|ABC\|_{\mathrm{F}} \le \|A\|_2\|B\|_{\mathrm{F}}\|C\|_2$ and (iii) $\langle A, B \rangle \le \|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}$.

For a symmetric matrix M, define $\mathrm{psd}(M)$ to be the closest positive semi-definite matrix to M in Frobenius norm. It can be shown that if $M = \sum_i \lambda_i v_i v_i^{\mathrm{T}}$, then $\mathrm{psd}(M) = \sum_{i:\lambda_i \ge 0} \lambda_i v_i v_i^{\mathrm{T}}$.

Weyl's Inequality. For matrices A and B, Weyl's inequality gives that $\sigma_{i+j-1}(A+B) \le \sigma_i(A) + \sigma_j(B)$ for all i and j. In particular, if $\|A - B\|_2 \le \varepsilon$, $|\sigma_i(A) - \sigma_i(B)| \le \varepsilon$ for all i.

Polynomials and Matrices. Let $p(x) = \sum_{i=0}^{d} p_i x^i$ be a degree d polynomial. We define $\|p\|_1 \coloneqq \sum_i |p_i|$ to be the sum of absolute values of the coefficients of the polynomial $p(x)$. Given $A \in \mathbb{R}^{n \times d}$, let $A = U\Sigma V^{\mathrm{T}}$ be the singular value decomposition of A with $\Sigma \in \mathbb{R}^{n \times d}$. Define $p(A) \coloneqq Up(\Sigma)V^{\mathrm{T}}$ where $p(\Sigma)$ is the matrix with main diagonal entries $p(\sigma_1), \ldots, p(\sigma_d)$. It is easy to check that the singular values of $p(A)$ are equal to $|p(\sigma_1)|, \ldots, |p(\sigma_d)|$.

Singular Value Excess. Let $A \in \mathbb{R}^{n \times d}$ with $n \ge d$ be an arbitrary matrix. Let $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_d \ge 0$ be the singular values of matrix A. The Singular Value Excess of matrix A, denoted by $\mathrm{sve}(A)$, is defined as the number of singular values of matrix A that are greater than or equal to 1 i.e.,

$$\mathrm{sve}(A) = |\{\, i \in [d] \mid \sigma_i \ge 1 \,\}|.$$

As eigenvalues of matrix $I - A^{\mathrm{T}}A$ are $1 - \sigma_1^2 \le \cdots \le 1 - \sigma_d^2$, $\mathrm{sve}(A)$ is equal to the number of non-positive eigenvalues of the matrix $I - A^{\mathrm{T}}A$. For any symmetric matrix M, let $k^-(M)$ denote the number of non-positive eigenvalues of the matrix M. For any matrix A, $\mathrm{sve}(A) = k^{-}(I - A^{\mathrm{T}}A)$.

Sketching Based Preconditioning for High-Precision Regression. Given a matrix $A \in \mathbb{R}^{n\times c}$ and a vector $b \in \mathbb{R}^n$, we use fast sketching based preconditioning methods given by the following theorem to obtain a $(1+\varepsilon)$ approximation to the problem $\min_x \|Ax - b\|_2$. See [Woo14] and references therein for more background.

Theorem 6.2.1 (High Precision Regression/Approximate Projections). *Given a matrix $A \in \mathbb{R}^{n\times c}$ and a vector $b \in \mathbb{R}^n$, we can compute a vector x in time $O((\mathrm{nnz}(A) + c^2)\log(1/\varepsilon) + c^{\omega})$ that satisfies $\|Ax - b\|_2^2 \le (1+\varepsilon)\|AA^{+}b - b\|_2^2$. By the Pythagorean theorem, the vector x obtained satisfies $\|AA^{+}b - Ax\|_2^2 \le \varepsilon\|AA^{+}b - b\|_2^2$.*

We have to pay c^{ω} only once to compute a preconditioner. Thereafter, every regression problem can be solved in time $O((\mathrm{nnz}(A) + c^2)\log(1/\varepsilon))$. We use $\textsc{HighPrecisionRegression}(A, b, \varepsilon)$ to denote the algorithm implied by Theorem 1. We extend the notation to compute approximate projections of each of the columns of matrix B, instead of just a single vector b, onto the column space of A.

Low Rank Approximation(LRA). Recall that the matrix $[A]_k$ optimally solves the problems $\min_{\text{rank-}k\ X} \|A - X\|_{\mathrm{F}}$ and $\min_{\text{rank-}k\ X} \|A - X\|_2$. As computing $[A]_k$ exactly is expensive, we use the Block Krylov Iteration algorithm of [MM15] to obtain a matrix $Z \in \mathbb{R}^{n\times k}$ for which $ZZ^{\mathrm{T}}A$ is a good solution to the Frobenius norm and spectral norm low rank approximation problems.

Theorem 6.2.2 ([MM15]). *Given a matrix $A \in \mathbb{R}^{n\times d}$ such that the products $Av \in \mathbb{R}^n$ and $A^{\mathrm{T}}v' \in \mathbb{R}^d$ can be computed in time T for any vectors $v \in \mathbb{R}^d$ and $v' \in \mathbb{R}^n$, the Block Krylov Iteration algorithm runs in time*

$$O\left(T\frac{k\log d}{\varepsilon^{1/2}} + \frac{nk^2\log^2(d)}{\varepsilon} + \frac{k^3\log^3(d)}{\varepsilon^{3/2}}\right)$$

and returns a matrix $Z \in \mathbb{R}^{n\times k}$ with orthonormal columns for which

$$\|A - ZZ^{\mathrm{T}}A\|_2 \le (1+\varepsilon)\|A - [A]_k\|_2 \text{ and } \|A - ZZ^{\mathrm{T}}A\|_{\mathrm{F}} \le (1+\varepsilon)\|A - [A]_k\|_{\mathrm{F}}.$$

Frobenius Norm Reduced-Rank Regression. As discussed in the introduction, there is a closed form solution to the reduced-rank Frobenius norm regression problem.

Lemma 6.2.3 (Lemma 4.1 of [Woo14], Lemma 2 of [MM15]). *Given matrices $A \in \mathbb{R}^{n\times c}$, $B \in \mathbb{R}^{n\times d}$, and a rank parameter $k \le c$, let matrix Q denote an orthonormal basis for the column span of A. Then $\min_{\text{rank-}k\ X} \|AX - B\|_{\mathrm{F}} = \|Q[Q^{\mathrm{T}}B]_k - B\|_{\mathrm{F}} = \|[AA^{+}B]_k - B\|_{\mathrm{F}}$. If $\bar{U}\bar{\Sigma}^2\bar{U}^{\mathrm{T}}$ is the SVD of $Q^{\mathrm{T}}AA^{\mathrm{T}}Q$, and $\bar{U}_k$ denotes the first k columns of $\bar{U}$, then $[Q^{\mathrm{T}}B]_k = \bar{U}_k\bar{U}_k^{\mathrm{T}}Q^{\mathrm{T}}B$, and therefore*

$$\min_{\text{rank-}k\ X} \|AX - B\|_{\mathrm{F}} = \|Q[Q^{\mathrm{T}}B]_k - B\|_{\mathrm{F}} = \|(Q\bar{U}_k)(Q\bar{U}_k)^{\mathrm{T}}B - B\|_{\mathrm{F}}.$$

Chebyshev Polynomials. The Chebyshev polynomials are defined as

$$T_0(x) = 1,\ T_1(x) = x \text{ and } T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$$

for all $i \geq 2$. Thus, $T_i(x)$ is a polynomial of degree i. It can be shown that if i is odd, then $T_i(x)$ has only odd degree monomials. Chebyshev polynomial T_i has the property that $\|T_i\|_1 \leq (1+\sqrt{2})^i$ for all i. See [MM15] for more properties of Chebyshev polynomials.

6.3 Previous work

Let $A \in \mathbb{R}^{n\times c}$ be a matrix and $U\Sigma V^{\mathrm{T}}$ be the "thin" SVD of A, where U is an orthonormal basis for the column space of A. Note that the projection matrix onto the column space of A is given by $AA^{+} = UU^{\mathrm{T}}$. The first algorithm to solve $\min_{\text{rank-}k\ X} \|AX - B\|_2$ was by Sou and Rantzer [SR12]. They consider the following problem:

$$\begin{aligned} &\text{minimize } \operatorname{rank}(X) \\ &\text{such that } \|AX - B\|_2 < 1. \end{aligned} \tag{6.1}$$

As multiplying a matrix with a projection matrix does not increase the operator norm, we have that $\|AX - B\|_2 \geq \|(I - AA^{+})(AX - B)\|_2 = \|(I - AA^{+})B\|_2$. Thus, the problem is feasible only when $\|(I - AA^{+})B\|_2 = \|(I - UU^{\mathrm{T}})B\|_2 < 1$. The following theorem characterizes the solution for (6.1).

Theorem 6.3.1 ([SR12]). *Given matrices $A \in \mathbb{R}^{n\times c}$ and a matrix $B \in \mathbb{R}^{n\times d}$, if there is a matrix Y such that $\|AY - B\|_2 < 1$, then the optimum value of (6.1) is sve(B) where sve(B) denotes the number of singular values of B that are greater than or equal to 1.*

For an arbitrary $s > 0$, consider the problem (6.1) with matrices A/s and B/s. The problem is feasible if and only if $\|(I - UU^{\mathrm{T}})(B/s)\|_2 < 1$, i.e., if and only if $\|(I - UU^{\mathrm{T}})B\|_2 < s$. Suppose s is such that $s > \|(I - UU^{\mathrm{T}})B\|_2$. Then Theorem 6.3.1 implies that there is a rank k matrix X such that $\|(A/s)X - (B/s)\|_2 < 1$ if and only if $k \geq \text{sve}(B/s)$, i.e., $\sigma_{k+1}(B/s) < 1$. This argument shows that for any $s > \max(\sigma_{k+1}(B), \|(I - UU^{\mathrm{T}})B\|_2)$, there is a rank k matrix X such that $\|AX - B\|_2 < s$. Thus, $\text{OPT} = \max(\sigma_{k+1}(B), \|(I - UU^{\mathrm{T}})B\|_2)$.

It is interesting and perhaps surprising that the above theorem implies we can obtain a solution that has a value $\max(\sigma_{k+1}(B), \|(I - UU^{\mathrm{T}})B\|_2)$, which is a simple lower bound on the optimum. This shows that if $\|(I - UU^{\mathrm{T}})B\|_2 \leq \sigma_{k+1}(B)$, there is a rank k matrix in the column span of matrix A that is as good of an approximation to B in spectral norm as $[B]_k$. Also, if $\|(I - UU^{\mathrm{T}})B\|_2 \geq \sigma_{k+1}(B)$, then there is a rank-$k$ matrix in the column space of A that is as good of an approximation to B in spectral norm as $AA^{+}B = UU^{\mathrm{T}}B$, the projection of B onto the column span of A.

We thus have the following corollary summarizing the discussion above. The corollary was also observed in [see Nam15, Section 4] in terms of a different parameter they call the critical rank.

Corollary 6.3.2. *Given matrices $A \in \mathbb{R}^{n\times c}$, $B \in \mathbb{R}^{n\times d}$ and a parameter k,*

$$\inf_{\text{rank-}k\ X} \|AX - B\|_2 = \max(\|(I - AA^+)B\|_2, \sigma_{k+1}(B)).$$

We give a proof of Theorem 6.3.1 for completeness in Appendix B.1.1. Our proof is similar to the proof in [SR12] with some minor changes.

6.4 Reduced-Rank Regression in Operator Norm

We first consider the case when c, d are small. In this case, we could assume that we can compute matrices U and Δ, where U is an orthonormal basis for the column span of matrix A, and the matrix $\Delta = B^{\mathrm{T}}(I - UU^{\mathrm{T}})B$. We give a simple algorithm that demonstrates our techniques. We then extend these ideas to the case when c, d are large, for which computing an orthonormal basis for A and computing Δ is prohibitively expensive.

From Corollary 6.3.2, we have that $\mathrm{OPT} = \max(\|(I - UU^{\mathrm{T}})B\|_2, \sigma_{k+1}(B))$. Let β be such that $(1+\varepsilon)\mathrm{OPT} \le \beta \le (1+2\varepsilon)\mathrm{OPT}$, which can be found using the Block Krylov algorithm. Throughout the chapter, we assume we know the value β.

Lemma 6.4.1. *If there exists a rank-k matrix X such that $\|UX - B\|_2 < \beta$, then $\sigma_{k+1}(U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}) < 1$.*

The proof of this lemma is in Appendix B.2.1. The proof of the above lemma also shows that if we can find a matrix Y of rank k such that $\|Y - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2 \le 1$, then we can obtain a matrix $X = Y(\beta^2 I - \Delta)^{1/2}$ such that $\|UX - B\|_2 < \beta$. Thus, we can compute the SVD of the matrix $U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}$ and obtain $[U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}]_k$ and obtain a solution $[U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}]_k(\beta^2 I - \Delta)^{1/2}$ of cost β.

Computing an exact SVD, as required in the proof of above Lemma, is much slower than computing a rank k matrix that satisfies the guarantees of the best rank k matrices approximately. The following lemma shows that we can obtain a solution of cost close to β even if we can compute a rank k matrix Y such that $\|Y - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2 \le 1 + \varepsilon$.

Lemma 6.4.2. *If Y is a rank k matrix such that $\|Y - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2 \le 1 + \varepsilon$, then we obtain that $\|UY(\beta^2 I - \Delta)^{1/2} - B\|_2 \le (1+\varepsilon)\beta$. Furthermore, $\|UY(UY)^+B - B\|_2 \le (1+\varepsilon)\beta$.*

The proof of this lemma is in Appendix B.2.2. The above lemma states that a $1+\varepsilon$ approximation to the best rank-k approximation of the matrix $U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}$ in operator norm is sufficient to find a solution of cost $(1+\varepsilon)\beta$ to the reduced-rank regression problem. We can use the Block Krylov algorithm to compute such an approximation. The Block Krylov algorithm of Musco and Musco [MM15] only needs an oracle to compute matrix-vector products. In the case when c, d are small, we can compute the matrices U, $(\beta^2 I - \Delta)^{-1/2}$ and then given arbitrary vectors v, v' we can compute $U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}v$ and $(\beta^2 I - \Delta)^{-1/2}B^{\mathrm{T}}Uv'$ and hence run the Block Krylov Algorithm.

Algorithm 6.1: Low Rank Approximation with Approximate Matrix Multiplication

```
Input: $M \in \mathbb{R}^{n\times d}, k \in \mathbb{Z}, \varepsilon > 0$, $\text{Oracle}_M : \mathbb{R}^d \times \varepsilon \to \mathbb{R}^n$, $\text{Oracle}_{M^{\mathrm{T}}} : \mathbb{R}^n \times \varepsilon \to \mathbb{R}^d$
Output: $Z \in \mathbb{R}^{n\times k}$
1 $G \sim \mathcal{N}(0,1)^{d\times k}, \kappa \leftarrow \sigma_1(M)/\sigma_{k+1}(M), q \leftarrow O((1/\sqrt{\varepsilon})\log(d/\varepsilon))$
2 $\varepsilon_\circ \leftarrow O\left(\min(\varepsilon/(\kappa^{2+5q}k^5d^2C^q)), \varepsilon^2/(48\kappa(\kappa^2(\sqrt{qk})k))\right)$
  /* Let $\circ$ denote approximate matrix-vector products using the Oracles
     with accuracy $\varepsilon_\circ$                                          */
3 $K' \leftarrow [(MM^{\mathrm{T}})^{\circ(q-1)/2}M \circ G, (MM^{\mathrm{T}})^{\circ(q-3)/2}M \circ G, \ldots, M \circ G]$
4 $Q' \leftarrow$ Orthonormal basis for $K'$
5 $[\bar{U}, \bar{\Sigma}^2, \bar{U}^{\mathrm{T}}] \leftarrow \mathrm{SVD}(Q'T(M \circ (M^{\mathrm{T}} \circ Q')))$
6 $\bar{U}_k \leftarrow$First $k$ columns of $\bar{U}$
7 $Z \leftarrow Q'\bar{U}_k$
```

This gives a $1 + O(\varepsilon)$ approximation to the reduced-rank regression problem.

When r, d are large, it is expensive to compute the matrices U, Δ and $(\beta^2 I - \Delta)^{-1/2}$. As the analysis in [MM15] works only when exact matrix-vector products can be computed, we cannot run the Block Krylov algorithm unless we compute the matrices U, Δ or at least are able to compute exact matrix vector products with the matrix $U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}$. So we analyze their algorithm and show that it works even using approximate matrix products instead of exact matrix products, given that the error is low enough.

6.5 Block Krylov Iteration with Approximate Multiplication Oracle

Given a parameter k and an oracle to approximately compute Mv and $M^{\mathrm{T}}v'$, given arbitrary vectors v and v', we would like to compute a matrix Z with k orthonormal columns such that

$$\|M - ZZ^{\mathrm{T}}M\|_2 \leq (1+\varepsilon)\sigma_{k+1}(M). \tag{6.2}$$

Specifically, suppose we have an oracle that, given an arbitrary vector v and approximation parameter $\varepsilon_\circ$, can compute in time $T(\varepsilon_\circ)$ a vector $M \circ v$ such that $\|Mv - (M \circ v)\|_2 \leq \varepsilon_\circ \|M\|_2 \|v\|_2$, and also given an arbitrary vector v' and accuracy parameter $\varepsilon_\circ$ can compute in time $T(\varepsilon_\circ)$ a vector $M^{\mathrm{T}} \circ v'$ such that $\|M^{\mathrm{T}}v' - M^{\mathrm{T}} \circ v'\|_2 \leq \varepsilon_\circ \|M\|_2 \|v'\|_2$. We are also given $\kappa = \sigma_1(M)/\sigma_{k+1}(M)$, and we want to compute a matrix Z as in (6.2).

Our algorithm to compute such a matrix Z is Algorithm 6.1. It is essentially the same as the Block Krylov algorithm of [MM15] with exact matrix-vector multiplication replaced by approximate matrix-vector multiplication with accuracy parameters as defined in our algorithm. Our main result for this section is the following theorem that states that the Block Krylov algorithm of [MM15] works even with approximate matrix-vector products.

Theorem 6.5.1. *Let $M \in \mathbb{R}^{n\times d}$, $k \leq d$ be a rank parameter, and $\varepsilon > 0$ be an accuracy parameter. Let $\kappa = \sigma_1(M)/\sigma_{k+1}(M)$. Given access to an oracle that can in time $T(\varepsilon_\circ)$ compute vectors $M \circ v$ and $M^{\mathrm{T}} \circ v'$ such that*

$$\|M \circ v - Mv\|_2 \leq \varepsilon_\circ \|M\|_2 \|v\|_2 \quad \text{and} \quad \|M^{\mathrm{T}} \circ v' - M^{\mathrm{T}} v'\|_2 \leq \varepsilon_\circ \|M\|_2 \|v'\|_2,$$

for any vectors v and v', Algorithm 6.1 computes a matrix $Z \in \mathbb{R}^{n\times k}$ with k orthonormal columns such that, with probability $\geq 3/5$, $\|(I - ZZ^{\mathrm{T}})M\|_2 \leq (1+\varepsilon)\sigma_{k+1}(M)$. The running time is

$$O\left(T\left(\frac{\varepsilon}{2\kappa^{5q}k^9 d^2 D^q}\right) qk + T\left(\frac{\varepsilon^2}{192\kappa^2(\sqrt{qk})k}\right) qk\right),$$

where $q = O\left((1/\sqrt{\varepsilon})\log(d/\varepsilon)\right)$ and D is an absolute constant. Further, if the approximations $M \circ v$ are spanned by M for all v, then the columns of the matrix Z are also spanned by the matrix M.

Proof Sketch. The proof of the Block Krylov algorithm of [MM15] first shows that there is a polynomial $p(x)$ that has only odd degree monomials such that the k-dimensional column space of the matrix $p(M)G$, where G is a Gaussian matrix with k columns, spans a $(1+\varepsilon)$ approximation. As we do not know how to compute this polynomial $p(x)$, the proof shows that the Krylov Space K spans this matrix $p(M)G$ and then shows that the rank k Frobenius norm approximation of the matrix M inside the Krylov subspace K is also a $1+\varepsilon$ spectral norm rank k approximation.

We adapt their proof to the case when we can compute matrix-vector products only approximately. We first show that the approximate Krylov matrix K' computed by Algorithm 6.1 is close to the actual Krylov matrix K in Lemma B.3.1. However, this lemma is not sufficient to directly prove that the rank-k Frobenius norm approximation of M inside the column space of K' is a $1+\varepsilon$ rank-k spectral approximation, since the matrices K and K' can be very poorly conditioned. Therefore, similar to the matrix $p(M)G$ in [MM15], we define a rank-k matrix Apx (see Equation B.3) and show that the matrix Apx is spanned by K'. Then we show in Lemma B.3.8 that the matrix Apx is close to $p(M)G$. Using an upper bound on the condition number of the matrix $p(M)G$ (see Lemma B.3.5), we conclude in Equation B.4 that the projection matrices onto the column spaces of the matrices Apx and $p(M)G$ are close to each other.

Similar to the argument of [MM15], we encounter the issue that this matrix Apx cannot be computed as we do not know the parameters of the polynomial $p(x)$, but we do have that this matrix Apx is spanned by the column space of K'. Using this fact, we show that an approximate rank k Frobenius norm approximation of M in the column space of K' is also a $1+\varepsilon$ spectral norm rank k approximation for the matrix M. We also show that this approximate rank k Frobenius norm approximation can be computed using approximate matrix-vector product oracles.

6.6 Approximate Oracles and Reduced Rank Regression

Lemma 6.4.2 shows that if Y is a rank k matrix such that $\|Y - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2 \le 1 + \varepsilon$, then $\|UY(UY)^{+}B - B\|_2 \le (1+\varepsilon)\beta$. Based on this result, we prove the following lemma which shows that a low rank-approximation of the matrix $AA^{+}B(\beta^2 I - \Delta)^{-1/2}$ suffices.

Lemma 6.6.1. *Let $\widetilde{Z} \in \mathbb{R}^{n\times k}$ be a matrix with orthonormal columns such that*

$$\|AA^{+}B(\beta^2 I - \Delta)^{-1/2} - \widetilde{Z}\widetilde{Z}^{\mathrm{T}}AA^{+}B(\beta^2 I - \Delta)^{-1/2}\|_2 \le 1 + \varepsilon.$$

Then $\|(AA^{+}\widetilde{Z})(AA^{+}\widetilde{Z})^{+}B - B\|_2 \le (1+\varepsilon)\beta$.

The proof of the lemma is in Appendix B.4.1. Hence, if we can get a good k-dimensional space $\widetilde{Z}$ for approximating the matrix $AA^{+}B(\beta^2 I - \Delta)^{-1/2}$, we can then obtain a good k dimensional space for B. We first show that we can instead find a low rank approximation for a matrix $AA^{+}BM/\beta$, for a suitable matrix M, which will also be a good low rank approximation for $AA^{+}B(\beta^2 I - \Delta)^{-1/2}$.

Lemma 6.6.2. *Given $\beta \ge (1+\varepsilon)\mathrm{OPT}$, there exists a polynomial $r(x)$ of degree at most $t = O\left(1/\sqrt{\varepsilon}\log(\kappa/\varepsilon)\right)$ such that for $M = r(\Delta/\beta^2)$, if $\widetilde{Z}$ is a matrix such that*

$$\|AA^{+}BM/\beta - \widetilde{Z}\widetilde{Z}^{\mathrm{T}}(AA^{+}BM/\beta)\|_2 \le 1 + \varepsilon,$$

then $\|AA^{+}B(\beta^2 I - \Delta)^{-1/2} - \widetilde{Z}\widetilde{Z}^{\mathrm{T}}AA^{+}B(\beta^2 I - \Delta)^{-1/2}\|_2 \le 1 + O(\varepsilon)$. Furthermore, $\|r\|_1 = O((1+\sqrt{2})^{O(\sqrt{1/\varepsilon}\log(\kappa/\varepsilon))}\log(\kappa/\varepsilon)/\varepsilon)$, $\|M\|_2 \le 2/\sqrt{\varepsilon}$, and $\sigma_{\min}(M) \ge 1/2$.

The proof of the above lemma is in Appendix B.4.2. From Theorem 6.5.1, to find a $1+\varepsilon$ approximation for rank k spectral norm low rank approximation (LRA) of the matrix $\mathcal{M}'$, we need only a way to compute the products $\mathcal{M}'v$ and $\mathcal{M}'^{\mathrm{T}}v'$ for any vectors v, v'. As $r(\Delta/\beta^2)$ is a polynomial in the matrix Δ/β^2, it is much easier to design approximate multiplication oracles for the matrix $AA^{+}BM/\beta$ than for the matrix $AA^{+}B(\beta^2 I - \Delta)^{-1/2}$. The following lemma shows that we can compute good approximations to the matrix vector products and then compute a $1+\varepsilon$ approximation to the LRA of matrix $\mathcal{M}' = AA^{+}B\frac{r(\Delta/\beta^2)}{\beta}$.

Lemma 6.6.3. *Given arbitrary vectors v, v' and an accuracy parameter ε_{f}, Algorithms B.1 and B.2 compute vectors y, y' such that $\|\mathcal{M}'v - y\|_2 \le \varepsilon_{\mathrm{f}}\|v\|_2$ and $\|\mathcal{M}'^{\mathrm{T}}v' - y'\|_2 \le \varepsilon_{\mathrm{f}}\|y\|_2$ in time*

$$\begin{aligned} T(\varepsilon_{\mathrm{f}}) &\coloneqq O(t\cdot(\mathrm{nnz}(B) + (\mathrm{nnz}(A)+c^2)\log\left(\kappa(B)^2\|r\|_1/(\varepsilon_{\mathrm{f}}\varepsilon)\right))) \\ &\quad + O((\mathrm{nnz}(A)+c^2)\log(\kappa(B)/(\varepsilon_{\mathrm{f}}\varepsilon))) \end{aligned}$$

where $t = O(\sqrt{1/\varepsilon}\log(\kappa/\varepsilon))$ and $\|r\|_1 = (1+\sqrt{2})^{O(1/\sqrt{\varepsilon}\log(\kappa/\varepsilon))}\log(\kappa/\varepsilon)/\varepsilon$.

Algorithm 6.2: Operator Norm Regression

```
    Input: $A \in \mathbb{R}^{n \times c}, B \in \mathbb{R}^{n \times d}, k \in \mathbb{Z}, \varepsilon > 0$
    Output: $X' \in \mathbb{R}^{c \times k}, X'' \in \mathbb{R}^{k \times d}$
1   $\beta \leftarrow (1 + \varepsilon/2) \max(\sigma_{k+1}(B), \|(I - AA^{+})B\|_2)$
2   $\Delta \leftarrow B^{\mathrm{T}}(I - AA^{+})B$                /* Not computed explicitly */
    /* Let $r(x)$ be the polynomial given by Lemma 6.6.2                  */
3   $\mathcal{M}' \leftarrow (AA^{+}B/\beta) r(\Delta/\beta^2)$                /* Not computed explicitly */
4   $Z \leftarrow$ Algorithm 6.1$(\mathcal{M}', k, \varepsilon/2,$ ApxProduct, ApxProdcutTranspose$)$
5   $X' \leftarrow$ HighPrecisionRegression$(A, Z, 1/2)$
6   $X'' \leftarrow Z^{\mathrm{T}} \cdot B$
```

6.6.1 Main Theorem

We finally have our main theorem that shows that Algorithm 6.2 outputs a $1 + \varepsilon$ approximation in factored form. The proof of the theorem is in Appendix B.4.4.

Theorem 6.6.4. *Given matrices $A \in \mathbb{R}^{n \times c}$ and $B \in \mathbb{R}^{n \times d}$, a rank parameter $k \leq c$ and an accuracy parameter ε, Algorithm 6.2 runs in time*

$$O\left(\left(\frac{\mathrm{nnz}(B) \cdot k}{\varepsilon} + \frac{\mathrm{nnz}(A) \cdot k}{\varepsilon^{1.5}} + \frac{c^2 k}{\varepsilon^{1.5}}\right) \cdot \mathrm{polylog}(\kappa, \kappa(AA^{+}B), d, k, 1/\varepsilon) + c^{\omega}\right),$$

and with probability $4/5$ outputs a matrix Z with k orthonormal columns, for which $\mathrm{colspan}(Z) \subseteq \mathrm{colspan}(A)$, such that $\|ZZ^{\mathrm{T}}B - B\|_2 \leq (1 + \varepsilon)\mathrm{OPT}$. It also outputs matrices $X' \in \mathbb{R}^{c \times k}$ and $X'' \in \mathbb{R}^{k \times d}$ such that $\|A(X' \cdot X'') - B\|_2 = \|ZZ^{\mathrm{T}}B - B\|_2 \leq (1 + \varepsilon)\mathrm{OPT}$.

6.6.2 Removing $\kappa(AA^{+}B)$ Dependence

We observe that we can add a random rank $k+1$ matrix to B to obtain a matrix $\widetilde{B}$ for which $\kappa(AA^{+}\widetilde{B})$ is bounded in terms of $\kappa(B)$. We also show that any arbitrary vector v can be multiplied with the matrix $\widetilde{B}$ in time comparable to $\mathrm{nnz}(B)$.

Lemma 6.6.5. *Given any matrices $A \in \mathbb{R}^{n \times c}$ and $B \in \mathbb{R}^{n \times d}$, if $\mathrm{rank}(A) \geq k+1$, then there exists a matrix $\widetilde{B}$ such that if*

$$\|A\widetilde{X} - \widetilde{B}\|_2 \leq (1 + \varepsilon/2) \min_{\text{rank-}k\ X} \|AX - \widetilde{B}\|_2 \tag{6.3}$$

for a rank k matrix $\widetilde{X}$, then

$$\|A\widetilde{X} - B\|_2 \leq (1 + \varepsilon)\mathrm{OPT}.$$

Additionally, $\kappa(AA^{\dagger}\widetilde{B}) = \sigma_1(AA^{+}\widetilde{B})/\sigma_{k+1}(AA^{+}\widetilde{B}) \leq (Cn/\varepsilon)\sigma_1(B)/\sigma_{k+1}(B)$, and given a vector v, $\widetilde{B}v$ can be computed in $O(\mathrm{nnz}(B) + (n + d)k)$ time.

The proof of this lemma is in Appendix B.4.5. Therefore, we run Algorithm 6.2 on matrix $\widetilde{B}$ and can compute a $(1+\varepsilon)$-approximate solution to the problem $\min_{\text{rank-}k\ X} \|AX - B\|_2$ in time

$$O\left(\left(\frac{\text{nnz}(B)\cdot k}{\varepsilon} + \frac{(n+d)k^2}{\varepsilon} + \frac{\text{nnz}(A)\cdot k}{\varepsilon^{1.5}} + \frac{c^2 k}{\varepsilon^{1.5}}\right)\cdot \text{polylog}(\kappa, n, d, k, 1/\varepsilon) + c^{\omega}\right). \tag{6.4}$$

6.7 Conclusions and Open Questions

In this work, we obtain fast algorithms for the reduced-rank regression with operator norm error. One of the key ingredients is showing that the Block Krylov iteration algorithm of Musco and Musco [MM15] does not need exact matrix products and that accurate matrix products are sufficient. But to satisfy the accuracy requirements in our algorithm, we require polynomial in $1/\varepsilon$ and k bits of precision.

The main open question is to determine if Block Krylov iteration algorithm, when run with only polylogarithmic bits of precision, computes accurate solutions. In a recent work [KW24], using the stability analysis of Musco, Musco, and Sidford [MMS18] for the Lanczos algorithm, we show that the LazySVD algorithm of Allen-Zhu and Li [AZL16] for computing Schatten-p norm low rank approximation (for $p \geq 2$) can be implemented with only polylogarithmic bits of precision. To our knowledge, this is the first stable algorithm for low rank approximation that can be implemented in time $O(\text{nnz}(A) \cdot k \cdot \text{polylog}(n)/\sqrt{\varepsilon})$. But block based algorithms are significantly fast on modern computing architectures and therefore resolving the stability of block Krylov iteration algorithm on floating point machines is still an important problem.

Chapter 7

Optimal Deterministic Coresets for Ridge Regression

7.1 Introduction

Linear least squares regression is one of the most popular tools for fitting a linear hypothesis to a given data set and ridge regression is an important regularized variant. When the number n of data points is very large, an intriguing question is whether there exists a small weighted subset of the data points which represents the entire data well for ridge regression. These subsets are often called *coresets*.

Let A be an $n \times d$ input matrix and B an $n \times d'$ matrix of labels corresponding to the data points in A. Here each label is a d'-dimensional vector. Let $a_i \in \mathbb{R}^d$ denote the i-th row of A. In the multiple-response least squares regression problem, the goal is to find a matrix X such that $\|AX - B\|_{\mathrm{F}}^2$ is minimized, where for a matrix C. In the ridge regression problem, we additionally add an ℓ_2-regularizer to the cost and now the goal is to find a matrix X which minimizes $\|AX - B\|_{\mathrm{F}}^2 + \lambda\|X\|_{\mathrm{F}}^2$, where $\lambda > 0$ is the regularization parameter.

We call a subset $S \subseteq [n]$, along with corresponding weights $w_i \geq 0$ for $i \in S$, an $\varepsilon-$coreset if the solution to the ridge regression problem

$$\widetilde{X}_{S,w} = \arg\min_X \sum_{i \in S} w_i \|a_i^T X - b_i\|_2^2 + \lambda \|X\|_{\mathrm{F}}^2$$

is a $(1+\varepsilon)$-approximate solution to the ridge regression problem $\min_X \|AX - B\|_{\mathrm{F}}^2 + \lambda\|X\|_{\mathrm{F}}^2 = \min_X \sum_i \|a_i^T X - b_i\|_2^2 + \lambda\|X\|_{\mathrm{F}}^2$ i.e.,

$$\begin{aligned} &\|A\widetilde{X}_{S,w} - B\|_{\mathrm{F}}^2 + \lambda\|\widetilde{X}_{S,w}\|_{\mathrm{F}}^2 \\ &\leq (1+\varepsilon)\left(\min_X \|AX - B\|_{\mathrm{F}}^2 + \lambda\|X\|_{\mathrm{F}}^2\right). \end{aligned}$$

Ideally, we would like to have the size $|S|$ of S be independent of n and depend linearly on the dimension of the data d and sublinear in $1/\varepsilon$. In the case of ridge regression, it is often desirable to have bounds in terms of the *statistical dimension* sd_λ of the input (defined below), which is always at most d and often significantly smaller than d.

Obtaining small subsets which accurately represent the entire data set is crucial for data interpretation and for efficient communication protocols. Note that unlike other solutions, such as directly computing the covariance matrix, coresets preserve the sparsity of the data. Indeed, if the rows of A and B are sparse, then the selected rows in the coreset are also sparse. As we will see, small coresets are extremely useful in giving efficient communication protocols to solve problems in a distributed setting.

In this work we focus on *deterministic* algorithms, i.e., algorithms with zero error probability. Since coresets are often composed multiple times in distributed protocols, this is desirable so that the error probability does not compound. Deterministic algorithms are automatically robust to adversarial inputs. In the distributed setting, the data at some servers could be a function of the coresets at other servers and the guarantees in this chapter will continue to hold.

7.1.1 Previous Work

There is a vast body of work on least squares and ridge regression, and we only touch upon the works most relevant to ours here and refer the reader to the surveys [Mah11, Woo14] and references therein. There is a long line of work on randomized sampling algorithms for speeding up least squares regression, see, e.g., [DKM06a, DKM06b, DKM06c, DMMS11]. Since our focus here is on deterministic algorithms, these are not directly useful for us. In the unregularized case, a direct technique that we can apply is the deterministic spectral sparsification result of Batson, Spielman, and Srivastava (BSS) [BSS12]. There are also several followup works [ALO15, LS18, CP19], but they give randomized rather than deterministic algorithms.

Assume $d' = O(\mathsf{sd}_\lambda) \leq d$. The issue with directly using the BSS algorithm for ordinary least squares regression is that naïvely one would need a so-called subspace embedding of the column span of $C = [A, B]$, the matrix with the columns of B adjoined to those of A. Consequently, this would result in a coreset S containing $O(d/\varepsilon^2)$ rows, which is larger than the $O(d/\varepsilon)$ that we desire. We instead achieve $O(d/\varepsilon)$ rows by combining the deterministic guarantees needed for regression in [ACW17] with a deterministic row selection algorithm achieving approximate matrix product in [CNW15]. Using this property, we can then bootstrap from it to in turn obtain a coreset of size $O(\mathsf{sd}_\lambda/\varepsilon)$. Directly applying techniques in [ACW17] would instead result in a coreset containing $O(\mathsf{sd}_\lambda/\varepsilon^2)$ rows.

Previous work [MJF19] has also observed that one can preserve the covariance matrix $C^T C$ exactly by a coreset of $O(d^2)$ rows by using Caratheodory's theorem, which can be implemented in deterministic polynomial time. However, it was not known if there is a matching lower bound in the case of least squares regression. There are strong lower bounds for cut and spectral sparsifiers [ACK+16, CKST17]; however, they fail to apply to the case of regression when there is a specific B matrix given.

There is also a body of work on distributed regression, for which each of the rows of $C = [A, B]$ reside on a single server. We refer the reader to the recent work [VWW19] and references therein. As shown in [VWW19], for ordinary least squares regression, $\Theta(d^2)$ words of communication per server is necessary and sufficient to solve the problem up to any relative error accuracy. The protocol is simple - each server computes its local covariance matrix and sends it to the coordinator, which can then solve the least squares problem exactly. While [VWW19] proves this is optimal, even to obtain a constant factor approximation, it need not be optimal if each row of C only has $O(1)$ non-zero entries. In this case one could hope to do better than d^2 communication by transmitting a small number of rows. We note that by Caratheordory's theorem, one can still transmit $O(d^2)$ rows or $O(\mathsf{sd}_\lambda^2)$ rows for the regularized version, assuming $d' \leq \mathsf{sd}_\lambda$, but the hope is to do even better. Alternatively, one can transmit a subspace embedding using $O(d/\varepsilon^2)$ rows, or $O(\mathsf{sd}_\lambda/\varepsilon^2)$ rows for the regularized version, but these are not linear in $1/\varepsilon$. Thus, an interesting question arises if there is a deterministic protocol achieving better communication. To the best of our knowledge, such work has not considered the sparse case, i.e., when each row of A and corresponding row of B have at most $O(1)$ non-zero entries.

7.1.2 Our Contributions

Given matrices $A \in \mathbb{R}^{n\times d}, B \in \mathbb{R}^{n\times d'}$ and parameter λ, we give a *deterministic* algorithm to find an ε-coreset S of size $O((\mathsf{sd}_\lambda(A) + d')/\varepsilon)$ and corresponding weights. We do this by using Corollary 1 from [CNW15] on suitably defined matrices and show that the matrix S thus obtained defines an ε-coreset for the ridge regression problem. This immediately gives that, with parameter $\lambda = 0$, there is an ε-coreset of size $O((\mathrm{rank}(A) + d')/\varepsilon)$.

Theorem 7.1.1. *Given matrices $A \in \mathbb{R}^{n\times d}, B \in \mathbb{R}^{n\times d'}$ and $\lambda \geq 0$, there exists a matrix S which selects and scales $O((\mathsf{sd}_\lambda + d')/\varepsilon)$ rows of A such that solution to the ridge regression problem*

$$\min_X \|SAX - SB\|_2^2 + \lambda\|X\|_{\mathrm{F}}^2$$

is a $(1 + \varepsilon)$ approximate solution to the ridge regression problem

$$\min_x \|AX - B\|_{\mathrm{F}}^2 + \lambda\|X\|_{\mathrm{F}}^2.$$

Using ε-coresets, we give an efficient communication protocol for computing a $1+\varepsilon$ approximate solution to multi-response ridge regression in a distributed setting with communication complexity of $O(s \cdot t \cdot (\min(s \cdot \mathsf{sd}_\lambda(A), \mathrm{rank}(A)) + d')/\varepsilon)$ words where s is the number of servers and t is the maximum number of non-zero elements in a row of $[A, B]$. In the case of $t \ll d$, this protocol is much more efficient than d^2 words which corresponds to naïvely sending the matrices $A_i^T A_i$ to the central server.

Theorem 7.1.2. *If rows of matrix $A \in \mathbb{R}^{n\times d}$ are partitioned among s servers and corresponding rows of $B \in \mathbb{R}^{n\times d'}$ are partitioned too, then there is a deterministic communication protocol using*

$$O(\frac{s \cdot t \cdot (\min(s \cdot \mathsf{sd}_\lambda(A), \operatorname{rank}(A)) + d')}{\varepsilon}) \text{ words,}$$

where t is the maximum number of non-zero entries in a row of $[A, B]$.

We finally show that our bounds on the coreset size are tight in the case of Multiple Ridge Regression for a certain setting of λ.

Theorem 7.1.3. *For all ε such that $1 \le 1/100\varepsilon \le d$ and $\lambda \le 1/4\varepsilon$, there exist matrices $A, B \in \mathbb{R}^{d/100\varepsilon \times d}$ for which any matrix S that selects and rescales k rows of A and B such that the solution to*

$$\min_X \|SAX - SB\|_F^2 + \lambda\|X\|_F^2$$

is a $(1+\varepsilon)$-approximation to

$$\min_X \|AX - B\|_F^2 + \lambda\|X\|_F^2$$

has $k = \Omega(\mathsf{sd}_\lambda(A)/\varepsilon)$ rows.

7.1.3 Notation

A typical ridge regression problem is given by inputs $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$ and $\lambda \ge 0$. Let $a_i \in \mathbb{R}^d$ be the vector corresponding to the i-th row of matrix A and $b_i \in \mathbb{R}$ be the i-th component of vector b. Let x^* denote the optimum solution for ridge regression and define $\mathrm{OPT} = \|Ax^* - b\|_2^2 + \lambda\|x^*\|_2^2$. A set $S \subseteq [n]$ along with weights $w_i \ge 0$ for $i \in S$ defines the weighted ridge regression problem

$$\min_x \sum_{i\in S} w_i(a_i^T x - b_i)^2 + \lambda\|x\|_2^2.$$

Let $\widetilde{x}_{S,w}$ be the optimal solution for the ridge regression problem defined by S, w. We say (S, w) is an ε-coreset if

$$\|A\widetilde{x}_{S,w} - b\|_2^2 + \lambda\|\widetilde{x}_{S,w}\|_2^2 \le (1+\varepsilon)\mathrm{OPT}.$$

For notational convenience, we define a selecting and scaling matrix S corresponding to set $S \subseteq n$ and w, such that

$$\|SAx - Sb\|_2^2 = \sum_{i\in S} w_i(a_i^T x - b_i)^2.$$

By a selecting matrix, we mean that each row of S has exactly one non-zero entry.

7.1.4 Preliminaries

Singular Value Decomposition

Statistical Dimension

Definition 7.1.4 (Statistical Dimension)**.** For a matrix $A \in \mathbb{R}^{n\times d}$ with non-zero singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$, the statistical dimension with respect to $\lambda \geq 0$, $\mathsf{sd}_\lambda(A)$, is defined to be

$$\mathsf{sd}_\lambda(A) = \sum_{i=1}^{\mathrm{rank}(A)} \frac{1}{1+\frac{\lambda}{\sigma_i^2}} \tag{7.1}$$

Wherever A is apparent, we use the notation sd_λ for $\mathsf{sd}_\lambda(A)$. Note that $\mathsf{sd}_\lambda(A) \leq \mathrm{rank}(A) \leq d$. This definition of statistical dimension captures our intuitive notion that as λ increases, the importance of the data decreases. Furthermore, if $\lambda \geq \sigma_1^2/\varepsilon$, then the vector 0 is a $1+\varepsilon$ approximate solution for any ridge regression problem with data matrix A and regularization parameter λ (See Lemma 14 of [ACW17] for a proof).

Below we note some properties of the statistical dimension we use in our proofs.

Lemma 7.1.5 (Lemma 12 of [ACW17])**.** *If $\hat{A}$ is a matrix with orthonormal columns such that*

$$\mathrm{range}(\hat{A}) = \mathrm{range}(\begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix})$$

and if U_1 comprises the first n rows of $\hat{A}$, then $\|U_1\|_\mathrm{F}^2 = \mathsf{sd}_\lambda(A)$ and $\|U_1\|_2^2 = 1/(1+\lambda/\sigma_1^2) \leq 1$.

The following lemmas follow directly from the definition of the statistical dimension and the fact that the singular values of a sub-matrix are dominated by the singular values of the whole matrix.

Lemma 7.1.6. *If A' is the sub-matrix of A formed by taking rows of A, then $\mathsf{sd}_\lambda(A') \leq \mathsf{sd}_\lambda(A)$.*

Lemma 7.1.7. *For any $r \geq 1$, $\mathsf{sd}_{\lambda/r}(A) \leq \min(r \cdot \mathsf{sd}_\lambda(A), \mathrm{rank}(A))$.*

Spectral Sparsification

Theorem 7.1.8. *(BSS Algorithm [BSS12]) Given n vectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^d$, there exists a subset $S \subseteq [n]$ of size $O(d/\varepsilon^2)$ with corresponding weights $w_i \geq 0$ for $i \in S$ such that*

$$(1-\varepsilon)\sum_{i=1}^n v_i v_i^T \preceq \sum_{i\in S} w_i v_i v_i^T \preceq (1+\varepsilon)\sum_{i=1}^n v_i v_i^T$$

and there is a deterministic polynomial time algorithm to find this subset along with the corresponding weights.

7.2 Upper Bounds for Linear Regression

In this section, we show that there exists an ε-coreset of size $O(d/\varepsilon)$ for "linear regression" in the single response case (when X has one column) and $O((d+d')/\varepsilon)$ in the multiple response case and show that the BSS algorithm can be used to find this coreset *deterministically*.

7.2.1 Single Response Linear Regression

Lemma 7.2.1 (Lemma 1 of [CNW15])**.** *If S is an ε-subspace embedding for* $\mathtt{colspan}(A, B)$*,*

$$\|A^{\mathrm{T}}S^{T}SB - A^{\mathrm{T}}B\|_2 \le \varepsilon\|A\|_2\|B\|_2.$$

Theorem 7.2.2. *If S is a $\sqrt{\varepsilon/4}$ subspace embedding for* $\mathtt{colspan}([A, b])$*, then $\widetilde{x}_{\text{OPT}} = \arg\min_x \|SAx - Sb\|_2^2 = (SA)^{+}(Sb)$ is a $(1+\varepsilon)$-approximate solution for the regression problem $\min_x \|Ax - b\|_2^2$.*

Proof. The proof goes along the line of [Sar06]. Let $A = U\Sigma V^{\mathrm{T}}$ be the "thin" singular value decomposition of A. Define $x_{\text{OPT}} = \arg\min_x \|Ax - b\|_2^2$. Using the Pythagorean theorem, we have

$$\|A\widetilde{x}_{\text{OPT}} - b\|_2^2 = \|Ax_{\text{OPT}} - b\|_2^2 + \|A\widetilde{x}_{\text{OPT}} - Ax_{\text{OPT}}\|_2^2 = \text{OPT} + \|A(SA)^{+}(Sb) - AA^{+}b\|_2^2.$$

Note that $AA^{+}b = UU^{\mathrm{T}}b$ and $(SA)^{+} = ((SA)^{\mathrm{T}}(SA))^{-1}(SA)^{\mathrm{T}} = (V\Sigma U^{\mathrm{T}}S^{\mathrm{T}}SU\Sigma V^{\mathrm{T}})^{-1}V\Sigma U^{\mathrm{T}}S^{\mathrm{T}} = V\Sigma^{-1}(U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1}U^{\mathrm{T}}S^{\mathrm{T}}$. Using these, we can write

$$\begin{aligned}\|A(SA)^{+}(Sb) - AA^{+}b\|_2^2 &= \|U(U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1}U^{\mathrm{T}}S^{\mathrm{T}}Sb - UU^{\mathrm{T}}b\|_2^2 \\ &= \|(U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1}U^{\mathrm{T}}S^{\mathrm{T}}Sb - U^{\mathrm{T}}b\|_2^2.\end{aligned}$$

We now observe that $\|U^{\mathrm{T}}S^{\mathrm{T}}SU - U^{\mathrm{T}}U\|_2 \le \sqrt{\varepsilon/4}$ which implies that $\sigma_{\min}(U^{\mathrm{T}}S^{\mathrm{T}}SU) \ge 1 - \sqrt{\varepsilon/4}$ and therefore,

$$\begin{aligned}\|(U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1}U^{\mathrm{T}}S^{\mathrm{T}}Sb - U^{\mathrm{T}}b\|_2^2 &\le \frac{1}{(1-\sqrt{\varepsilon/4})^2}\|U^{\mathrm{T}}S^{\mathrm{T}}Sb - (U^{\mathrm{T}}S^{\mathrm{T}}SU)U^{\mathrm{T}}b\|_2^2 \\ &\le \frac{1}{(1-\sqrt{\varepsilon/4})^2}\|U^{\mathrm{T}}S^{\mathrm{T}}S(b - UU^{\mathrm{T}}b)\|_2^2.\end{aligned}$$

Using the fact that $\|U^{\mathrm{T}}(b - UU^{\mathrm{T}}b)\|_2 = 0$ and Lemma 7.2.1, we conclude that

$$\frac{1}{(1-\sqrt{\varepsilon/4})^2}\|U^{\mathrm{T}}S^{\mathrm{T}}S(b - UU^{\mathrm{T}}b)\|_2^2 = \frac{1}{(1-\sqrt{\varepsilon/4})^2}\|U^{\mathrm{T}}S^{\mathrm{T}}S(b - UU^{\mathrm{T}}b) - U^{\mathrm{T}}(b - UU^{\mathrm{T}}b)\|_2^2$$

$$\leq \frac{1}{(1-\sqrt{\varepsilon/4})^2}\cdot(\varepsilon/4)\cdot\|U\|_2^2\|b - UU^{\mathrm{T}}b\|_2^2 \leq \varepsilon\cdot\mathrm{OPT}$$

since $\|b - UU^{\mathrm{T}}b\|_2^2 = \mathrm{OPT}$. We therefore have $\|A\widetilde{x}_{\mathrm{OPT}} - b\|_2^2 \leq (1+\varepsilon)\cdot\mathrm{OPT}$. □

Theorem 7.2.3. *Given matrix $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, there exists a matrix S which selects $O(d/\varepsilon)$ rows of A and scales them such that solution to the regression problem* $\arg\min_x \|SAx - Sb\|_2^2$ *is a $(1+\varepsilon)$-approximation to the regression problem $\|Ax - b\|_2^2$. This implies the existence of an $O(d/\varepsilon)$-sized coreset.*

Proof. Applying BSS to the matrix $[A, b]$ with parameter $O(\sqrt{\varepsilon})$ gives a selecting and rescaling matrix S with $O(d/\varepsilon)$ rows such that S is a $\sqrt{\varepsilon/4}$ subspace embedding for $\mathrm{colspan}(A, b)$. By Theorem 7.2.2, we get that the solution to the regression problem $\min_x \|SAx - Sb\|_2^2$ is a $(1+\varepsilon)$ approximate solution to the problem $\min_x \|Ax - b\|_2^2$. □

Theorem 7.2.4. *Given matrix $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, there exists a matrix S which selects $O(d^2)$ rows of A and scales them such that the solution to the regression problem* $\arg\min_x \|SAx - Sb\|_2^2$ *is an optimal solution to* $\arg\min_x \|Ax - b\|_2^2$.

Proof. The proof of this theorem is similar to that of [MJF19], and is included here for completeness. Assume that the matrix A is full rank. Let a_i be the i^{th} row of A written as a column. Let $\widetilde{a}_i \in \mathbb{R}^{d+1}$ be the vector a_i appended with b_i. Consider the matrices $\widetilde{a}_i\widetilde{a}_i^T$ for $i = 1\ldots n$. The matrix $(1/n)\sum_{i=1}^n \widetilde{a}_i\widetilde{a}_i^T$ lies in the convex hull of the matrices $\widetilde{a}_i\widetilde{a}_i^T$ for $i = 1\ldots n$. By Caratheodory's theorem, there exists a set $\mathcal{S} \subseteq [n]$, $|\mathcal{S}| = O(d^2)$ and corresponding weights $w_i \geq 0$ for $i \in \mathcal{S}$, such that

$$\frac{1}{n}\sum_{i=1}^n \widetilde{a}_i\widetilde{a}_i^T = \sum_{j\in\mathcal{S}} w_j\widetilde{a}_j\widetilde{a}_j^T$$

We obtain the following relations from the above:

$$\sum_{i=1}^n a_ia_i^T = \sum_{j\in\mathcal{S}}(nw_j)\, a_ja_j^T$$

$$\sum_{i=1}^n b_ia_i = \sum_{j\in\mathcal{S}}(nw_j)\, b_ja_j$$

Let $s_1, s_2, \ldots, s_{|\mathcal{S}|} \in [n]$ be the elements of $\mathcal{S}$. Define a sampling and rescaling matrix S as follows

$$S_{i,s_i} = \sqrt{nw_{s_i}} \; i = 1 \ldots |\mathcal{S}|$$

and the rest of the entries of S are 0s. Then,

$$\begin{aligned}
&\arg\min_x \|SAx - Sb\|_2^2 = (A^{\mathrm{T}} S^T SA)^{-1}(A^{\mathrm{T}} S^T Sb) \\
&= \left(\sum_{i=1}^{|\mathcal{S}|} S_{i,s_i}^2 a_{s_i} a_{s_i}^T\right)^{-1} \left(\sum_{i=1}^{|\mathcal{S}|} S_{i,s_i}^2 b_{s_i} a_{s_i}\right) \\
&= \left(\sum_{j\in\mathcal{S}} nw_j \, a_j a_j^T\right)^{-1} \left(\sum_{j\in\mathcal{S}} nw_j \, b_j a_j\right) \\
&= \left(\sum_{i=1}^{n} a_i a_i^T\right)^{-1} \left(\sum_{i=1}^{n} b_i a_i\right) \\
&= (A^{\mathrm{T}} A)^{-1}(A^{\mathrm{T}} b) \\
&= \arg\min_x \|Ax - b\|_2^2
\end{aligned}$$

□

7.2.2 Multiple Response Linear Regression

We now consider the problem of multiple response linear regression, where given matrices $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times d'}$, we find the solution of the following optimization problem

$$\min_{X\in\mathbb{R}^{d\times d'}} \|AX - B\|_{\mathrm{F}}^2.$$

Theorem 7.2.5. *Given matrices $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times d'}$, if the matrix S is a $\sqrt{\varepsilon/4}$ subspace embedding for* colspan(A, B)*, then the solution to the optimization problem*

$$\widetilde{X} = \arg\min_{X\in\mathbb{R}^{d\times d'}} \|SAX - SB\|_{\mathrm{F}}^2$$

is a $1 + \varepsilon$ approximate solution to the multiple response regression problem on matrices A, B, i.e.,

$$\|A\widetilde{X} - B\|_{\mathrm{F}}^2 \le (1+\varepsilon) \min_{X\in\mathbb{R}^{d\times d'}} \|AX - B\|_{\mathrm{F}}^2.$$

Such a matrix S with $O((d + d')/\varepsilon)$ rows can be obtained using BSS.

Proof. Let x_i denote the i-th column of X and b_i be the i-th column of matrix B. Then the multiple response linear regression can be written as

$$\min_{x_1,x_2,\ldots,x_{d'}} \sum_i \|Ax_i - b_i\|_2^2.$$

These are d' independent single response linear regression problems and given that S is a $\sqrt{\varepsilon/4}$ subspace embedding for $\mathtt{colspan}(A,B)$, we get that for all $i = 1\ldots d'$, S is a $\sqrt{\varepsilon/4}$ subspace embedding for $\mathtt{colspan}(A,b_i)$. From Theorem 7.2.2, $\widetilde{x_i} = \min_x \|SAx - Sb_i\|_2^2$ is a $1+\varepsilon$ approximate solution to the regression problem on A and b_i and hence,

$$\sum_{i=1}^{d'} \|SA\widetilde{x_i} - b_i\|_2^2 \leq \sum_{i=1}^{d'} (1+\varepsilon)\min_{x_i} \|Ax_i - b_i\|_2^2.$$

So, the matrix $\widetilde{X}$ having ith column equal to $\widetilde{x_i}$ is a $(1+\varepsilon)$ approximate solution for the regression problem on (A,B). Thus,

$$\|A\widetilde{X} - B\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\min_X \|AX - B\|_{\mathrm{F}}^2.$$ □

7.3 Upper Bounds for Ridge Regression: Statistical Dimension

In this section, we extend our results to the case of ridge regression and present coresets of size $O((\mathsf{sd}_\lambda + d')/\varepsilon)$. We use approximate matrix product techniques of Cohen, Nelson and Woodruff [CNW15] to obtain the bounds in terms of statistical dimension.

Theorem 7.3.1. *Given matrices $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{n\times d'}$ and $\lambda \geq 0$, there exists a matrix S which selects and scales $O((\mathsf{sd}_\lambda + d')/\varepsilon)$ rows of A such that solution to the ridge regression problem*

$$\min_X \|SAX - SB\|_2^2 + \lambda\|X\|_{\mathrm{F}}^2$$

is a $(1+\varepsilon)$ approximate solution to the ridge regression problem

$$\min_X \|AX - B\|_{\mathrm{F}}^2 + \lambda\|X\|_{\mathrm{F}}^2.$$

Proof. Consider the matrix $\hat{A} = \begin{bmatrix} A \\ \sqrt{\lambda} I_d \end{bmatrix}$ and $\hat{B} = \begin{bmatrix} B \\ 0_{d\times d'} \end{bmatrix}$. Let $\mathscr{A}$ be a matrix with orthonormal columns such that $\mathtt{range}(\mathscr{A}) = \mathtt{range}([\hat{A}\,\hat{B}])$ and the first d columns of $\mathscr{A}$ are a basis for $\mathtt{colspan}(\hat{A})$. Let

$$\mathscr{A} = \begin{bmatrix} \mathcal{U}_1 \\ \mathcal{U}_2 \end{bmatrix} = \begin{bmatrix} U_1 & U_1' \\ U_2 & U_2' \end{bmatrix}$$

where $\mathcal{U}_1 \in \mathbb{R}^{n\times(d+d')}$, $\mathcal{U}_2 \in \mathbb{R}^{d\times(d+d')}$, $U_1 \in \mathbb{R}^{n\times d}$, $U_1' \in \mathbb{R}^{n\times d'}$, $U_2 \in \mathbb{R}^{d\times d}$ and $U_2 \in \mathbb{R}^{d\times d'}$. We have $\|\mathcal{U}_1\|_2^2 = \|[U_1\ U_1']\|_2^2 \leq 1$ and using Lemma 7.1.5 we get

$$\|\mathcal{U}_1\|_F^2 = \|[U_1\ U_1']\|_F^2 = \|U_1\|_F^2 + \|U_1'\|_F^2 \leq \mathsf{sd}_\lambda(A) + d'.$$

By Corollary 1 of [CNW15], we can obtain a *selecting* and *scaling* matrix S with $O((\mathsf{sd}_\lambda + d')/(\varepsilon/16)) = O((\mathsf{sd}_\lambda + d')/\varepsilon)$ rows such that

$$\begin{aligned}\|\mathcal{U}_1^T S^T S \mathcal{U}_1 - \mathcal{U}_1^T \mathcal{U}_1\|_2^2 &\leq \frac{\varepsilon}{16}\left(\|\mathcal{U}_1\|_2^2 + \frac{\|\mathcal{U}_1\|_F^2}{\mathsf{sd}_\lambda(A) + d'}\right)^2 \\ &\leq \frac{\varepsilon}{16}\left(1 + \frac{\mathsf{sd}_\lambda(A) + d'}{\mathsf{sd}_\lambda(A) + d'}\right)^2 \\ &= \varepsilon/4\end{aligned}$$

Consider the *selecting* and *scaling* matrix

$$\mathcal{S} = \begin{bmatrix} S & 0 \\ 0 & I \end{bmatrix}$$

We have $\|\mathcal{A}^T\mathcal{S}^T\mathcal{S}\mathcal{A} - \mathcal{A}^T\mathcal{A}\|_2^2 = \|\mathcal{U}_1^T S^T S\mathcal{U}_1 + \mathcal{U}_2^T\mathcal{U}_2 - \mathcal{U}_1^T\mathcal{U}_1 - \mathcal{U}_2^T\mathcal{U}_2\|_2^2 = \|\mathcal{U}_1^T S^T S\mathcal{U}_1 - \mathcal{U}_1^T\mathcal{U}_1\|_2^2 \leq \varepsilon/4$. Hence, $\mathcal{S}$ is a $\sqrt{\varepsilon/4}$ subspace embedding for $\mathtt{range}(\mathcal{A}) = \mathtt{range}[\hat{A}\ \hat{B}]$. By Theorem 7.2.5, we have that the solution to the regression problem

$$\begin{aligned}\min_X \|\mathcal{S}\hat{A}X - \mathcal{S}\hat{B}\|_F^2 &= \min_X \left\| \begin{bmatrix} SA \\ \sqrt{\lambda} I \end{bmatrix} X - \begin{bmatrix} SB \\ 0 \end{bmatrix} \right\|_F^2 \\ &= \min_X \|SAX - SB\|_F^2 + \lambda\|X\|_F^2\end{aligned}$$

is a $(1+\varepsilon)$ approximate solution to

$$\begin{aligned}&\min_X \|\hat{A}X - \hat{B}\|_F^2 = \min_X \left\| \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} X - \begin{bmatrix} B \\ 0 \end{bmatrix} \right\|_F^2 \\ &= \min_X \|AX - B\|_F^2 + \lambda\|X\|_F^2\end{aligned}$$

□

7.4 Deterministic Communication Protocol for Ridge Regression

7.4.1 Communication Model

We consider the communication model in which there are s servers and there is a central coordinator which can communicate with every server. All communication occurs through two-way communication channels between the servers and the coordinator. The coordinator initiates the communication protocol and always decides who speaks next. This model is simpler to analyze and can simulate the arbitrary peer-to-peer communication model with a communication complexity of at most twice that of the peer-to-all model, by instead of having server A talk directly to server B, having server A forward its message through the coordinator. We must also add $\log s$ bits per message to tell the coordinator who to forward the message to.

7.4.2 Ridge Regression in the Distributed Setting

Consider the setting of ridge regression in a row-partition distributed setting. Let there be s servers with matrices $A_1, A_2, \ldots, A_s$ and corresponding label matrices $B_1, B_2, \ldots, B_s$, respectively. Let A be the matrix obtained by stacking $A_1, A_2, \ldots, A_s$ and B be the matrix obtained by stacking $B_1, B_2, \ldots, B_s$. Assume that ε and all the entries are multiples of $1/\text{poly}(nd)$ and are upper bounded by $\text{poly}(nd)$. Therefore, by multiplying all the entries by $\text{poly}(nd)$, we can assume that all the entries are integers and are upper bounded by $\text{poly}(nd)$ and hence each entry takes $O(\log(nd))$ bits to encode. This assumption also ensures that all the weights evaluated can be rounded to be encoded using $O(\log(nd)) + O(\log(1/\varepsilon))$ bits. We call $O(\log(nd))$ bits a *word*. Let there be a central coordinator each server can communicate with. We would like to compute a $(1+\varepsilon)$ approximate solution to the following optimization problem while minimizing the communication required

$$\begin{aligned}&\min_X \|AX - B\|_F^2 + \lambda\|X\|_F^2 \\ &= \min_X \left(\sum_{i=1}^{s} \|A_iX - B_i\|_F^2\right) + \lambda\|X\|_F^2\end{aligned}$$

Theorem 7.4.1. *If rows of matrix $A \in \mathbb{R}^{n\times d}$ are partitioned among s servers and corresponding rows of $B \in \mathbb{R}^{n\times d'}$ are partitioned too, then there is a deterministic communication protocol using*

$$O\left(\frac{s\cdot t\cdot(\min(s\cdot \mathsf{sd}_\lambda(A), \text{rank}(A)) + d')}{\varepsilon}\right) \text{ words,}$$

where t is the maximum number of non-zero entries in a row of $[A, B]$.

Proof. For each server i, define the following matrices

$$\hat{A}_i = \begin{bmatrix} A_i \\ \sqrt{\lambda/s}\cdot I_d \end{bmatrix} \qquad \hat{B}_i = \begin{bmatrix} B_i \\ 0 \end{bmatrix}$$

Let $\mathcal{A}_i$ be a matrix with orthonormal columns such that $\texttt{range}(\mathcal{A}_i) = \texttt{range}([\hat{A}_i\ \hat{B}_i])$. From the proof of Theorem 7.3.1, we obtain a $\sqrt{\varepsilon/4}$ subspace embedding $\mathcal{S}_i$ of the form $\begin{bmatrix} S_i & 0 \\ 0 & I \end{bmatrix}$ for $\texttt{range}(\mathcal{A}_i) =$

$\texttt{range}([\hat{A}_i\ \hat{B}_i])$. The matrix S_i selects and scales $O((\mathsf{sd}_{\lambda/s}(A_i) + d')/\varepsilon)$ rows of A_i and B_i. Now, we have that the matrix

$$\mathcal{S} = \texttt{diag}(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_s)$$

is a $\sqrt{\varepsilon/4}$ subspace-embedding for

$$\texttt{range}(\begin{bmatrix} \mathcal{A}_1 \\ \vdots \\ \mathcal{A}_s \end{bmatrix}) = \texttt{range}(\begin{bmatrix} \hat{A}_1\ \hat{B}_1 \\ \vdots\ \ \vdots \\ \hat{A}_s\ \hat{B}_s \end{bmatrix}).$$

From Theorem 7.2.5, we obtain that solution to the optimization problem

$$\begin{aligned}
\min_X \|\mathcal{S} \begin{bmatrix} \hat{A}_1 \\ \vdots \\ \hat{A}_s \end{bmatrix} X - \mathcal{S} \begin{bmatrix} \hat{B}_1 \\ \vdots \\ \hat{B}_s \end{bmatrix} \|_F^2 &= \min_X \sum_i \|\mathcal{S}_i \hat{A}_i X - \mathcal{S}_i \hat{B}_i\|_F^2 \\
&= \min_X \sum_i (\|S_i A_i x - S_i B_i\|_F^2 + \frac{\lambda}{s}\|X\|_F^2) \\
&= \min_X (\sum_i \|S_i A_i X - S_i B_i\|_F^2) + \lambda\|X\|_F^2 \\
&= \min_X \|SAX - b\|_2^2 + \lambda\|X\|_2^2
\end{aligned}$$

where S is defined as $S = \texttt{diag}(S_1, S_2, \ldots, S_s)$ is a $(1 + \varepsilon)$ approximate solution to the regression problem.

$$\begin{aligned}
\min_X \| \begin{bmatrix} \hat{A}_1 \\ \vdots \\ \hat{A}_s \end{bmatrix} X - \begin{bmatrix} \hat{B}_1 \\ \vdots \\ \hat{B}_s \end{bmatrix} \|_F^2 &= \min_X \sum_i \|\hat{A}_i X - \hat{B}_i\|_F^2 \\
&= \min_X \sum_i (\|A_i X - B_i\|_F^2 + \frac{\lambda}{s}\|X\|_F^2) \\
&= \min_X (\sum_i \|A_i X - B_i\|_F^2) + \lambda\|X\|_F^2 \\
&= \min_X \|AX - B\|_F^2 + \lambda\|X\|_F^2.
\end{aligned}$$

The communication protocol is as follows: the i-th server computes the *selecting* and *scaling* matrix S_i as above and sends the matrices $S_i A_i$ and $S_i B_i$ to the central server (also called the coordinator). The central server can now compute the solution to the problem $\min_X \sum_i \|S_i A_i X - S_i B_i\|_F^2 + \lambda\|X\|_F^2$ by standard techniques. The solution obtained is guaranteed to be a $(1+\varepsilon)$ solution as shown above.

When each row of matrix $[A_i, B_i]$ has at most t non-zero entries, the communication required is at most $O(s \cdot t \cdot \frac{\max_i \mathsf{sd}_{\lambda/s}(A_i)+d'}{\varepsilon})$ words for the entries of the matrices and $O(s \cdot \frac{\max_i \mathsf{sd}_{\lambda/s}(A_i)+d'}{\varepsilon})$ words for the weights. But, for all i, $\mathsf{sd}_{\lambda/s}(A_i) \le \mathsf{sd}_{\lambda/s}(A) \le \min(s \cdot \mathsf{sd}_\lambda(A), \operatorname{rank}(A))$. Hence, the communication complexity is $O(\frac{s \cdot t \cdot (\min(s \cdot \mathsf{sd}_\lambda(A), \operatorname{rank}(A))+d')}{\varepsilon})$ words. □

7.5 Lower Bounds for Multi Response Ridge Regression

In this section, we give example matrices $A, B \in \mathbb{R}^{d/100\varepsilon \times d}$, $\varepsilon > 0, \lambda \ge 0$ such that $\mathsf{sd}_\lambda(A) = \Omega(d)$ and any *selecting* and *scaling* matrix S needs at least $\Omega(d/\varepsilon)$ rows for it to give a $1+\varepsilon$ approximate solution.

Theorem 7.5.1. *For all ε such that $1 \le 1/100\varepsilon \le d$ and $\lambda \le 1/4\varepsilon$ there exist matrices $A, B \in \mathbb{R}^{d/100\varepsilon \times d}$ for which any matrix S that selects and rescales k rows of A and B such that the solution to*

$$\min_X \|SAX - SB\|_F^2 + \lambda \|X\|_F^2$$

is a $(1+\varepsilon)$ approximation to $\min_X \|AX - B\|_F^2 + \lambda\|X\|_F^2$ has $k = \Omega(\mathsf{sd}_\lambda(A)/\varepsilon)$ rows.

Proof. Let the matrix A be a block matrix where each block has dimensions $1/100\varepsilon \times 1$. Define blocks on the diagonal of A to be the vectors $1_{1/100\varepsilon}$ and remaining entries of A to be 0. The singular values of this matrix are all equal to $\sqrt{1/100\varepsilon}$. For $\lambda \le 1/4\varepsilon$, $\mathsf{sd}_\lambda(A) \ge d/26$. Similarly, let B be a block matrix, where each block has size $1/100\varepsilon \times d$. So, the matrix B is formed by stacking matrices $B_1, B_2, \ldots, B_d$. Let each row of B_i be a distinct unit vector in the standard basis for $\mathbb{R}^d$. We can choose $1/100\varepsilon$ distinct standard basis vectors as $d \ge 1/100\varepsilon$. For the block matrix B_i, define a set $H_i \subseteq [d]$ of integers k such that e_k is a row in B_i.

The problem $\min_X \|AX - B\|_F^2 + \lambda\|X\|_F^2$ is equivalent to

$$\min_{x_1, x_2, \ldots, x_d} (\|1_{1/100\varepsilon} x_1^T - B_1\|_F^2 + \lambda\|x_1\|_2^2) + \ldots + (\|1_{1/100\varepsilon} x_d^T - B_d\|_F^2 + \lambda \|x_d\|_2^2) \tag{7.2}$$

and the above is equivalent to minimizing each of the problems independently.

We consider the problem

$$\min_{x_1} \|1_{1/100\varepsilon} x_1^T - B_1\|_F^2 + \lambda\|x_1\|_2^2. \tag{7.3}$$

Without loss of generality assume that $H_1 = [1/100\varepsilon]$ i.e., rows of B_1 are the first $1/100\varepsilon$ standard basis vectors of $\mathbb{R}^d$. Consider a matrix S which selects k rows of $[1_{1/100\varepsilon}, B_1]$ and solves the following optimization problem

$$\min_{x_1} \|S 1_{1/100\varepsilon} x_1^T - SB_1\|_F^2 + \lambda \|x_1\|_2^2$$

We can assume that S selects the top k rows without loss of generality. Then the above problem is equivalent to

$$\min_{y_1,y_2,\ldots,y_d\in\mathbb{R}} \sum_{i=1}^{k}(w_i(1-y_i)^2 + (W-w_i)y_i^2 + \lambda y_i^2) + \sum_{i=k+1}^{d}(\lambda+W)y_i^2$$

where w_i is the weight S assigns to error in row i and $W = \sum_{i=1}^{k} w_i$. By taking the partial derivative of the objective function with respect to y_i and setting it to 0, we get that it is minimized when $y_i = w_i/(W+\lambda)$ for $i = 1\ldots k$ and $y_i = 0$ for $i = k+1\ldots d$. When this solution is used for the original ridge regression problem (7.3), the cost is

$$\frac{1}{100\varepsilon} - k + \sum_{i=1}^{k}\left(1-\frac{w_i}{W+\lambda}\right)^2 + \left(1-\frac{1}{100\varepsilon}\right)\left(\frac{w_i}{W+\lambda}\right)^2 + \lambda\left(\frac{w_i}{W+\lambda}\right)^2$$

For a fixed W, the cost is minimized when all w_i's are equal and hence we set $w_i = W/k = b$ for some $b \geq 0$. The cost can now be written as

$$\begin{aligned}
&\frac{1}{100\varepsilon} - k + k\left[\left(1-\frac{b}{\lambda+kb}\right)^2 + \left(\frac{1}{100\varepsilon}-1\right)\left(\frac{b}{\lambda+bk}\right)^2 + \lambda\left(\frac{b}{\lambda+bk}\right)^2\right] \\
&= \frac{1}{100\varepsilon} - k + k\left[1 + c^2 - 2c + \left(\frac{1}{100\varepsilon}-1\right)c^2 + \lambda c^2\right] \quad \text{where } c = \frac{b}{\lambda+bk} \\
&= \frac{1}{100\varepsilon} - k + k + k\left[\left(\frac{1}{100\varepsilon}+\lambda\right)c^2 - 2c\right] \\
&= \frac{1}{100\varepsilon} + k\left[\left(\frac{1}{100\varepsilon}+\lambda\right)c^2 - 2c\right]. \qquad (7.4)
\end{aligned}$$

This is minimized when $c = \frac{1}{\lambda+1/100\varepsilon}$ which is obtained when $b = \frac{\lambda}{\lambda+(1/100\varepsilon-k)}$ (This is a valid setting of weights as $k \leq 1/100\varepsilon$ and hence $b \geq 0$). The minimum value is hence equal to $1/100\varepsilon - k/(\lambda + 1/100\varepsilon)$. This is the least error we can get on (7.3) using any matrix S which selects and re-scales $\leq$ k rows. Substituting $k = 1/100\varepsilon$ we recover the OPTvalue for (7.3) which is equal to $1/100\varepsilon - 1/(1+100\lambda\varepsilon)$. Now, $1/100\varepsilon - k/(\lambda+1/100\varepsilon)$ is $\leq (1+2\varepsilon)$OPTiff

$$\begin{aligned}
& & \frac{1}{100\varepsilon} - \frac{k}{\lambda+1/100\varepsilon} &\leq (1+2\varepsilon)\left(\frac{1}{100\varepsilon} - \frac{1}{1+100\lambda\varepsilon}\right) \\
&\text{iff} & \frac{1}{100\varepsilon} - \frac{k}{\lambda+1/100\varepsilon} &\leq \frac{1}{100\varepsilon} + \frac{1}{50} - \frac{1+2\varepsilon}{1+100\lambda\varepsilon} \\
&\text{iff} & -k &\leq \frac{\lambda+1/100\varepsilon}{50} - \frac{1+2\varepsilon}{100\varepsilon} \\
& & &= \frac{2\varepsilon\lambda + 1/50 - 1 - 2\varepsilon}{100\varepsilon}
\end{aligned}$$

$$\text{iff} \qquad k \geq \frac{49/50 - 2\varepsilon\lambda + 2\varepsilon}{100\varepsilon}.$$

For any $\lambda \leq 1/4\varepsilon$, this implies that $k \geq 1/400\varepsilon$.

To get a $(1+\varepsilon)$ approximate solution to (7.2), we need to solve at least $d/2$ sub-problems up to $(1+2\varepsilon)$ approximation. Hence, the selecting matrix S for the whole problem must select at least $d/2 \times 1/400\varepsilon = \Omega(d/\varepsilon) = \Omega(\mathsf{sd}_\lambda(A)/\varepsilon)$ rows. $\square$

This shows the $O(d^2)$ upper bound to construct covariance matrices using Caratheodory's theorem is tight.

Chapter 8

Sketching Algorithms and Lower Bounds for Ridge Regression

8.1 Introduction

Given a matrix $A \in \mathbb{R}^{n \times d}$, a vector $b \in \mathbb{R}^n$, and a parameter $\lambda \geq 0$, recall that the ridge regression problem is defined as:

$$\min_x \|Ax - b\|_2^2 + \lambda\|x\|_2^2.$$

Throughout this chapter, we assume $n \leq d$, and we let x^* be the optimal solution for the problem. Let OPT be the optimal value of the ridge regression problem. In this setting, an earlier work [CYD18] gives an iterative algorithm using subspace embeddings. The following theorem states their results when their algorithm is run for one iteration. Note that their algorithm is more general and when run for t iterations, the error is proportional to ε^t.

Theorem 8.1.1 (Theorem 1 of [CYD18]). *Given $A \in \mathbb{R}^{n \times d}$, let $V \in \mathbb{R}^{d \times n}$ be an orthonormal basis for the rowspace of matrix A. If $S \in \mathbb{R}^{m \times d}$ is a matrix which satisfies*

$$\|V^{\mathrm{T}} S^{\mathrm{T}} S V - I_n\|_2 \leq \varepsilon/2, \tag{8.1}$$

then $\widetilde{x} = A^{\mathrm{T}}(AS^{\mathrm{T}}SA^{\mathrm{T}} + \lambda I_n)^{-1} b$ satisfies

$$\|\widetilde{x} - x^*\|_2 \leq \varepsilon\|x^*\|_2.$$

Recall that a matrix S which satisfies (8.1) is called an $\varepsilon/2$ subspace embedding for the column space of V, since for any y in colspan(V), we have $(1 - \varepsilon/2)\|y\|_2^2 \leq \|Sy\|_2^2 \leq (1 + \varepsilon/2)\|y\|_2^2$. There are many *oblivious* and *non-oblivious* constructions of subspace embeddings (see Chapter 2). Recall that the *oblivious* subspace embedding (OSE) constructions do not depend on the matrix V that is to be embedded. OSEs specify a distribution $\mathcal{S}$ such that for any arbitrary matrix V, a random matrix

S drawn from the distribution $\mathcal{S}$ is an ε subspace embedding for V with probability $\geq 1 - \delta$. On the other hand, non-oblivious constructions compute a distribution $\mathcal{S}$ that depends on the matrix V that is to be embedded.

In many cases, such as in streaming, it is important that the sketch used is *oblivious*, since matrix-dependent subspace embedding constructions may need to read the entire input matrix first. Oblivious sketches also allow turnstile updates to the matrix A in a stream. In the turnstile model of streaming, we receive updates of the form $((i, j), v)$ which update $A_{i,j}$ to $A_{i,j} + v$. In this chapter, we focus on algorithms for ridge regression that use *oblivious* sketching matrices.

To satisfy (8.1), using CountSketch [CW17, MM13], we can obtain a sketching dimension of $m = O(n^2/\varepsilon^2)$ for which the matrix SA^{T} can be computed in $O(\mathrm{nnz}(A))$ time, where $\mathrm{nnz}(A)$ denotes the number of nonzero entries in the matrix A. Using OSNAP embeddings [NN13, Coh16], we can obtain a sketching dimension of $m = O(n^{1+\gamma}\log(n)/\varepsilon^2)$ for which the matrix SA^{T} can be computed in time $O(\mathrm{nnz}(A)/\gamma\varepsilon)$. For $\gamma = O(1/\log(n))$, we have $m = O(n\log(n)/\varepsilon^2)$ with SA^{T} that can be computed in time $O(\mathrm{nnz}(A)\log(n)/\varepsilon)$. We can see that there is a trade-off between CountSketch and OSNAP — one has a smaller sketching dimension while the other is faster to apply to a given matrix. If $t_{SA^{\mathrm{T}}}$ is the time required to compute SA^{T}, then $\widetilde{x}$ in Theorem 8.1.1 can be computed in time $O(\mathrm{nnz}(A) + t_{SA^{\mathrm{T}}} + mn^{\omega-1} + n^{\omega})$ where ω is the matrix multiplication constant. Thus, it is important to have both a small $t_{SA^{\mathrm{T}}}$ and small m to obtain fast running times.

When allowed $O(\log(1/\varepsilon))$ passes over the input matrix A, the algorithm of Chowdhury, Yang and Drineas [CYD18] produces an ε relative error solution using only a constant, say $1/2$ subspace embedding. When only $O(1)$ passes are allowed over the input, their algorithm requires a $\delta = f(\varepsilon)$ subspace embedding to obtain ε error solutions. As seen above this leads to either a high value of m or a high value of $t_{SA^{\mathrm{T}}}$.

We show that we only need a simpler Approximate Matrix Multiplication (AMM) guarantee, along with a constant subspace embedding, instead of requiring S to be an $\varepsilon/2$ subspace embedding.

Definition 8.1.2 (AMM). Given matrices A and B of appropriate dimensions, a matrix S satisfies the ε-AMM property for (A, B) if

$$\|A^{\mathrm{T}}S^{\mathrm{T}}SB - A^{\mathrm{T}}B\|_{\mathrm{F}} \leq \varepsilon\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}.$$

We now state the guarantees of our algorithm (Algorithm 8.1) for 1 iteration, requiring 2 passes over the matrix A.

Theorem 8.1.3. *If S is a random matrix such that for any fixed $d \times n$ orthonormal matrix V and a vector r, with probability $\geq 9/10$,*

$$\|V^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}V - I_n\|_2 \leq 1/2$$

and

$$\|V^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}Vr - r\|_2 \leq (\varepsilon/2\sqrt{n})\|V\|_{\mathrm{F}}\|Vr\|_2 = (\varepsilon/2)\|r\|_2, \tag{8.2}$$

then $\widetilde{x} = A^{\mathrm{T}}(AS^{\mathrm{T}}SA^{\mathrm{T}} + \lambda I_n)^{-1}b$ *satisfies* $\|\widetilde{x} - x^*\|_2 \le \varepsilon\|x^*\|_2$ *with probability* $\ge 9/10$.

We show that the **OSNAP** distribution satisfies both of these two properties with a sketching dimension of $r = O(n\log(n) + n/\varepsilon^2)$ and with SA^{T} that can be computed in $O(\mathrm{nnz}(A)\cdot\log(n))$ time. Note that our algorithm (Algorithm 8.1) is also more general, and when run for t iterations, the error is proportional to ε^t. Our algorithm differs from that of [CYD18] in that our algorithm needs a fresh sketching matrix in each iteration whereas their algorithm only needs one sketching matrix across iterations.

Many natural problems in the streaming literature have been studied specifically with two passes [CKP+21, KN21, AR20, BW11]. Also in the case of federated learning, where minimizing the number of rounds of communication is important [PHK+21], the smaller sketch sizes required by our algorithm (Algorithm 8.1) gives an improvement over the algorithm of [CYD18].

We can also bound the cost of $\widetilde{x}$ computed by our algorithm. For any $x \in \mathbb{R}^d$, let $\mathrm{cost}(x) = \|Ax - b\|_2^2 + \lambda\|x\|_2^2$. Bounds on $\|\widetilde{x} - x^*\|_2$ also let us obtain an upper bound on $\mathrm{cost}(\widetilde{x})$. It can be shown that for any vector x, $\mathrm{cost}(\widetilde{x}) = \mathrm{OPT} + \|A(x^* - \widetilde{x})\|_2^2 + \lambda\|x^* - \widetilde{x}\|_2^2$. Thus, $\|\widetilde{x} - x^*\|_2 \le \varepsilon\|x^*\|_2$ implies that $\mathrm{cost}(\widetilde{x}) = \mathrm{OPT} + (\sigma^2 + \lambda)\varepsilon^2\|x^*\|_2^2 \le (1 + (1 + \sigma^2/\lambda)\varepsilon^2)\mathrm{OPT}$. We are most interested in the case $\sigma^2 \ge \lambda$, as it is when $\mathrm{cost}(\widetilde{x})$ could be much higher than OPT. Setting $\varepsilon = O(\sqrt{\delta\lambda/\sigma^2})$, we obtain that the solution $\widetilde{x}$ returned by Theorem 8.1.3 is a $1 + \delta$ approximation.

We also show that our algorithm can be used to obtain approximate solutions to Kernel Ridge Regression with a polynomial kernel. We show that instantiating the construction of [AKK+20] with appropriate sketching matrices gives a fast way to apply sketches, satisfying the subspace embedding and AMM properties, to the matrix $\phi(A)$, where the i-th row of the matrix $\phi(A)$ is given by $A_{i*}^{\otimes p}$.

8.1.1 Lower bounds for Ridge Regression

It can be seen that the optimal solution $x^* = A^{\mathrm{T}}(AA^{\mathrm{T}} + \lambda I_n)^{-1}b$. Our algorithm, for one iteration, is simply to compute $\widetilde{x} = A^{\mathrm{T}}(AS^{\mathrm{T}}SA^{\mathrm{T}} + \lambda I_n)^{-1}b$ for a matrix S that satisfies the requirements in Theorem 8.1.3. All the algorithm does is substitute the expensive matrix product AA^{T}, which can take $O(n\cdot\mathrm{nnz}(A))$ time to compute, with the matrix product $AS^{\mathrm{T}}SA^{\mathrm{T}}$, which only takes $t_{SA^{\mathrm{T}}} + mn^{\omega-1}$ time to compute. Thus, constructing "good" distributions for which $\widetilde{x}$ is a $1+\varepsilon$ approximation seems to be the most natural way to obtain fast algorithms for ridge regression. As discussed previously, **OSNAP** matrices with $m = O(n\log(n) + n\sigma^2/\lambda\varepsilon)$ and having near-optimal $t_{SA^{\mathrm{T}}} = \widetilde{O}(\mathrm{nnz}(A))$ can be used to compute a solution $\widetilde{x}$ that is a $1+\varepsilon$ approximation. We show that for a large class of nice-enough distributions over $m\times d$ matrices $\mathcal{S}$, if $S \sim \mathcal{S}$ satisfies that $\widetilde{x} = A^{\mathrm{T}}(AS^{\mathrm{T}}SA^{\mathrm{T}} + \lambda I)^{-1}b$ is a $1+\varepsilon$ approximation with high probability, then $r = \Omega(n\sigma^2/\lambda\varepsilon)$. This shows that **OSNAP** matrices have both a near-optimal sketching dimension r and near-optimal time $t_{SA^{\mathrm{T}}}$. We show the lower bound by showing that for any "nice" distribution $\mathcal{S}$ for which $\widetilde{x}$ is a $1+\varepsilon$ approximation with high probability, the distribution must also satisfy an Approximate Matrix Multiplication (AMM) guarantee, i.e., for any matrix B, for $S \sim \mathcal{S}$, $\|B^{\mathrm{T}}S^{\mathrm{T}}SB - B^{\mathrm{T}}B\|_{\mathrm{F}}$ must be small with high probability. We then show a lower bound on m for any distribution $\mathcal{S}$ which satisfies the AMM guarantee. Here we demonstrate our techniques in the simple case of $n = 1$. Without loss of generality, we assume $\lambda = 1$.

Consider the ridge regression problem $\min_x (a^{\mathrm{T}}x - b)^2 + \|x\|_2^2$, where the vector $a \in \mathbb{R}^d$ is arbitrary. We have $\widetilde{x} = a(a^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}a + 1)^{-1}b$ and

$$\mathrm{cost}(\widetilde{x}) = \left(\frac{\|a\|_2^2 b}{\|\boldsymbol{S}a\|_2^2 + 1} - b\right)^2 + \frac{\|a\|_2^2}{(\|\boldsymbol{S}a\|_2^2 + 1)^2} b^2$$

whereas $\mathrm{OPT} = b^2/(\|a\|_2^2 + 1)$. For $\|a\|_2 \geq 100/\sqrt{\varepsilon}$, it turns out that unless $(1 - \sqrt{\varepsilon}/\|a\|_2)\|a\|_2^2 \leq \|\boldsymbol{S}a\|_2^2 \leq (1 + \sqrt{\varepsilon}/\|a\|_2)\|a\|_2^2$, we will have $\mathrm{cost}(\widetilde{x}) \geq (1 + \varepsilon/2)\mathrm{OPT}$. Thus, for $\widetilde{x}$ to be a $1 + \varepsilon/2$ approximation with probability $\geq 99/100$ for any arbitrary a, it must be the case that with probability $\geq 99/100$, $|a^{\mathrm{T}}a - a^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}a| = |\|a\|_2^2 - \|\boldsymbol{S}a\|_2^2| \leq (\sqrt{\varepsilon}/\|a\|_2)\|a\|_2^2$ i.e., $\boldsymbol{S}$ must satisfy the AMM property with parameter $\sqrt{\varepsilon}/\|a\|_2$. We show an $\Omega(1/\delta^2)$ lower bound for any distribution which satisfies the δ-AMM property, which gives a lower bound of $\Omega(\|a\|_2^2/\varepsilon)$ for ridge regression for $n = 1$.

For the case of general n, we show that any "nice" distribution $\mathcal{S}$ that gives $1 + \varepsilon$ approximate solutions for ridge regression must satisfy the $\sqrt{\varepsilon/n\sigma^2}$-AMM guarantee, which by using the lower bound for AMM, gives an $\Omega(n\sigma^2/\varepsilon)$ lower bound for ridge regression.

To prove the lower bound, we crucially use the fact that the sketching distribution $\mathcal{S}$ must satisfy that $\widetilde{x}$ is a $1+\varepsilon$ approximation for *any* particular ridge regression problem instance (A, b) with high probability.

8.1.2 Lower bounds for AMM

We prove the following lower bound for oblivious sketching matrices that give AMM guarantees.

Theorem 8.1.4 (Informal)**.** *If $\mathcal{S}$ is a distribution over $m \times d$ matrices such that for any $n \times d$ matrix A, $\boldsymbol{S} \sim \mathcal{S}$ satisfies with probability $\geq 99/100$, that*

$$\|A\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}A^{\mathrm{T}} - AA^{\mathrm{T}}\|_{\mathrm{F}} \leq \delta\|A\|_{\mathrm{F}}\|A^{\mathrm{T}}\|_{\mathrm{F}},$$

for $\delta \leq c/\sqrt{n}$, then $m = \Omega(1/\delta^2)$ where $c > 0$ is a small enough universal constant.

To the best of our knowledge, this is the first tight lower bound on the dimension of oblivious sketching matrices for AMM. The lower bound is tight up to constant factors as the CountSketch distribution with $m = O(1/\delta^2)$ rows has the above property. Note that for $\delta = \varepsilon/n$, the distribution $\mathcal{S}$ as in the above theorem satisfies that for any $d \times n$ orthonormal matrix V, with probability $\geq 99/100$,

$$\|V^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}V - I_n\|_{\mathrm{F}} \leq (\varepsilon/n)\|V\|_{\mathrm{F}}^2 = \varepsilon.$$

Thus, a distribution $\mathcal{S}$ that has the ε/n-AMM property also has the ε-subspace embedding property. [NN14] gives an $\Omega(n/\varepsilon^2)$ lower bound for such distributions, thus giving an $\Omega(1/(\delta^2 n))$ lower bound for δ-AMM for small enough δ. Our result gives a stronger $\Omega(1/\delta^2)$ lower bound.

We now give a brief overview of our proof for $n = 1$. Consider $a \in \mathbb{R}^d$ to be a fixed unit vector and let $\mathcal{S}$ be a distribution supported on $r \times d$ matrices as in the above theorem. Then we have $\mathbf{Pr}_{\boldsymbol{S}\sim\mathcal{S}}[|a^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}a-1| \le \delta] \ge 0.99$. Let $U\Sigma V^{\mathrm{T}}$ be the singular value decomposition of $\boldsymbol{S}$ with $\Sigma \in \mathbb{R}^{r\times r}$. Without loss of generality, we can assume that V^{T} is independent of Σ and that V^{T} is a uniformly random orthonormal matrix. This follows from the fact that if $\boldsymbol{S}$ is an *oblivious* AMM sketch, then $\boldsymbol{SQ}$ is also an *oblivious* AMM sketch, where $\boldsymbol{Q}$ is a uniformly random $d\times d$ orthogonal matrix independent of $\boldsymbol{S}$. Thus, we have $\mathbf{Pr}_{\Sigma,V^{\mathrm{T}}}[|a^{\mathrm{T}}V\Sigma^2V^{\mathrm{T}}a - 1| \le \delta] \ge 0.99$, where Σ and V^{T} are random matrices that correspond to the AMM sketch $\boldsymbol{S}$, as described.

Jiang and Ma [JM17] show that if $m = o(d)$, then the total variation distance between $V^{\mathrm{T}}a$ and $(1/\sqrt{d})\boldsymbol{g}$ is small, where $\boldsymbol{g}$ is an m dimensional vector with independent Gaussian entries. Thus, we obtain that $\mathbf{Pr}_{\Sigma,\boldsymbol{g}}[|(1/d)\boldsymbol{g}^{\mathrm{T}}\Sigma^2\boldsymbol{g} - 1| \le \delta] = \mathbf{Pr}_{\Sigma,\boldsymbol{g}}[|(1/d)\sum_{i=1}^m \sigma_i^2\boldsymbol{g}_i^2 - 1| \le \delta] \ge 0.95$.

If $\sum_{i=1}^m \sigma_i^2 \le d/200$, then $(1/d)\sum_{i=1}^m \sigma_i^2\boldsymbol{g}_i^2 \le 1/2$ with probability ≥ 0.99 by Markov's inequality. So, $\mathbf{Pr}_{\Sigma}[\sum_{i=1}^m \sigma_i^2 \le d/200]$ must be small. On the other hand, $\mathbf{Var}\left((1/d)\sum_i \sigma_i^2\boldsymbol{g}_i^2\right) = (2/d^2)\sum_{i=1}^m \sigma_i^4$. Thus, for $(1/d)\sum_{i=1}^m \sigma_i^2\boldsymbol{g}_i^2$ to concentrate in the interval $(1-\delta, 1+\delta)$, we would expect $\sqrt{\mathbf{Var}\left((1/d)\sum_i \sigma_i^2\boldsymbol{g}_i^2\right)} \approx \delta$, which implies $(2/d^2)\sum_{i=1}^m \sigma_i^4 \approx \delta^2$. Thus, with a reasonable probability, it must be simultaneously true that $d/200 \le \sum_{i=1}^m \sigma_i^2$ and $\sum_{i=1}^m \sigma_i^4 \approx d^2\delta^2/2$. Then,

$$d^2/(200)^2 \le \left(\sum_{i=1}^m \sigma_i^2\right)^2 \le m\sum_{i=1}^m \sigma_i^4 \approx md^2\delta^2/2,$$

thus obtaining $m \gtrsim \Omega(1/\delta^2)$. We extend this proof idea to the general case of $n \ge 1$.

Non-asymptotic upper bounds on the total variation (TV) distance between Gaussian matrices and sub-matrices of random orthogonal matrices obtained in recent works [JM17, LW21] let us replace the matrices that are harder to analyze with Gaussian matrices in our proof of the lower bound for AMM. We believe this technique could be helpful in proving tight lower bounds for other types of sketching guarantees.

8.2 Preliminaries

For arbitrary matrices M, N, the symbol t_{MN} denotes the time required to compute the product MN. We recall a few useful definitions.

Definition 8.2.1 (Approximate Matrix Multiplication). Given an integer d, we say that an $m \times d$ random matrix $\boldsymbol{S}$ has the (ε, δ)-AMM property if for any matrices A and B with d rows, we have that

$$\|A^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}B - A^{\mathrm{T}}B\|_{\mathrm{F}} \le \varepsilon\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}},$$

with probability $\ge 1-\delta$ over the randomness of $\boldsymbol{S}$.

We usually drop δ from the notation by picking it to be a small enough constant.

Definition 8.2.2 (Oblivious Subspace Embeddings)**.** Given an integer d, an $m \times d$ random matrix S is an (ε, δ)-OSE for n-dimensional subspaces if for any arbitrary $d \times n$ matrix A, with probability $\geq 1 - \delta$, simultaneously for all vectors x,

$$\|SAx\|_2^2 \in (1 \pm \varepsilon)\|Ax\|_2^2.$$

For both OSEs and distributions satisfying the (ε, δ)-AMM property, two major parameters of importance are the size of the sketch (m), and the time to compute SA (t_{SA}).

8.3 An Iterative Algorithm for Ridge Regression

The following theorem describes the guarantees of the solution $\hat{x}$ returned by Algorithm 8.1.

Theorem 8.3.1. *If Algorithm 8.1 samples independent sketching matrices $S_j \in \mathbb{R}^{m\times d}$ for all $j \in [t]$ satisfying the properties*

1. *with probability $\geq 1 - 1/(20t)$, for all vectors x, $\|S_j A^{\mathrm{T}} x\|_2^2 \in (1 \pm 1/2)\|A^{\mathrm{T}}x\|_2^2$, and*
2. *for all arbitrary matrices M, N, with probability $\geq 1 - 1/(20t)$,*

$$\|M^{\mathrm{T}} S_j^{\mathrm{T}} S_j N - M^{\mathrm{T}} N\|_{\mathrm{F}} \leq \sqrt{\varepsilon/4n}\|M\|_{\mathrm{F}}\|N\|_{\mathrm{F}},$$

then with probability $\geq 9/10$, $\|\hat{x} - x^*\|_2 \leq (\sqrt{\varepsilon})^t \|x^*\|_2$ and further $\mathrm{cost}(\hat{x}) \leq (1 + (\sigma^2/\lambda + 1)\varepsilon^t)\mathrm{OPT}$.

We prove a few lemmas which give intuition about the algorithm before proving the above theorem.

Algorithm 8.1: RidgeRegression

```
Input: $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^d, t \in \mathbb{Z}, \varepsilon, \lambda > 0$
Output: $\hat{x} \in \mathbb{R}^d$
1 $b^{(0)} \leftarrow b, \widetilde{x}^{(0)} \leftarrow 0_d, y^{(0)} \leftarrow 0_n$
2 for $j = 1, \dots, t$ do
3     $b^{(j)} \leftarrow b^{(j-1)} - \lambda y^{(j-1)} - A\widetilde{x}^{(j-1)}$
4     $S_j \leftarrow$ 1/2 subspace embedding for the rowspace of $A$ and has the $\sqrt{\varepsilon/4n}$ AMM property
5     $y^{(j)} \leftarrow (AS_j^{\mathrm{T}} S_j A^{\mathrm{T}} + \lambda I)^{-1} b^{(j)}$
6     $\widetilde{x}^{(j)} \leftarrow A^{\mathrm{T}} y^{(j)}$
7 end
8 $\hat{x} \leftarrow \sum_{j=1}^{t} \widetilde{x}^{(j)}$
9 return $\hat{x}$
```

After $i - 1$ iterations of the algorithm, $\sum_{j=1}^{i-1} \widetilde{x}^{(j)}$ is the estimate for the optimum solution x^*. At a high level, in the i-th iteration, the algorithm is trying to compute an approximation to the difference $x^* - \sum_{j=1}^{i-1} \widetilde{x}_j$ by computing an approximate solution to the problem

$$\min_x \|A(x + \sum_{j=1}^{i-1} \widetilde{x}_j) - b\|_2^2 + \lambda\|x + \sum_{j=1}^{i-1} \widetilde{x}_j\|_2^2.$$

Let $x^{*(j)} = A^{\mathrm{T}}(AA^{\mathrm{T}} + \lambda I)^{-1}b^{(j)}$. The following lemma shows that the solution to the above problem is $x^{*(i)}$.

Lemma 8.3.2. *For all i, $x^* = x^{*(i)} + \sum_{j=1}^{i-1} \widetilde{x}^{(j)}$.*

Proof. Let $f(x) = \|A(x + \sum_{j=1}^{i-1} \widetilde{x}^{(j)}) - b\|_2^2 + \lambda\|x + \sum_{j=1}^{i-1} \widetilde{x}^{(j)}\|_2^2$ and z be the solution realizing $\min_x f(x)$. We have $\nabla_x f(x)\,|_{x=z} = 0$ giving $z = (A^{\mathrm{T}}A + \lambda I)^{-1}(A^{\mathrm{T}}b - (A^{\mathrm{T}}A + \lambda I)\sum_{j=1}^{i-1} \widetilde{x}^{(j)})$.

Noting that $\widetilde{x}^{(j)} = A^{\mathrm{T}}y^{(j)}$ for all j and that for all i, $b^{(i)} = b - \lambda\sum_{j=1}^{i-1} y^{(j)} - \sum_{j=1}^{i-1} A\widetilde{x}^{(j-1)}$, we obtain that $z = (A^{\mathrm{T}}A + \lambda I)^{-1}A^{\mathrm{T}}b^{(i)}$. Now using the matrix identity $(A^{\mathrm{T}}A + \lambda I)^{-1}A^{\mathrm{T}} = A^{\mathrm{T}}(AA^{\mathrm{T}} + \lambda I)^{-1}$, we get $z = x^{*(i)}$ is the optimal solution to $\min_x f(x)$.

As $x^{*(i)}$ is the optimal solution, it is also clear that $x^* = x^{*(i)} + \sum_{j=1}^{i-1} \widetilde{x}^{(j)}$ since otherwise x^* is not the optimal solution for the original ridge regression problem, which is a contradiction. □

So by the end of the $(j-1)$-th iteration, the estimate to x^* is off by $x^{*(j)}$. The algorithm is approximating $x^{*(j)} = A^{\mathrm{T}}(AA^{\mathrm{T}} + \lambda I)^{-1}b^{*(j)}$ with $\widetilde{x}^{(j)} = A^{\mathrm{T}}(A\boldsymbol{S}_j^{\mathrm{T}}\boldsymbol{S}_jA^{\mathrm{T}} + \lambda I)^{-1}b^{*(j)}$. The following lemma gives the error of this approximation assuming that the sketching matrix $\boldsymbol{S}_j$ has both the subspace embedding and AMM properties. This is the part where our proof differs from that of the proof of [CYD18].

Lemma 8.3.3. *If $\boldsymbol{S}_j$ is drawn from a distribution such that for any fixed matrix A^{T}, $\boldsymbol{S}_j$ is a $1/2$ subspace embedding with probability $1-\delta$ and for any fixed matrices M, N, with probability $1-\delta$,*

$$\|M^{\mathrm{T}}\boldsymbol{S}_j^{\mathrm{T}}\boldsymbol{S}_jN - M^{\mathrm{T}}N\|_{\mathrm{F}} \le \sqrt{\varepsilon/n}\|M\|_{\mathrm{F}}\|N\|_{\mathrm{F}},$$

then with probability $\ge 1-2\delta$, $\|x^{*(j)} - \widetilde{x}^{(j)}\|_2 \le (2\sqrt{\varepsilon})\|x^{*(j)}\|_2$.

Proof. Let $A = U\Sigma V^{\mathrm{T}}$ be the thin singular value decomposition of A. Since we assume that $n \le d$, the matrix V^{T} has a size $n \times d$. We have $x^{*(j)} = V\Sigma(I + \Sigma^2)^{-1}U^{\mathrm{T}}b^{(j)}$. By using $(I + \Sigma^2)^{-1} = \Sigma^{-1}(I + \Sigma^{-2})^{-1}\Sigma^{-1}$, we get $x^{*(j)} = V(I + \Sigma^{-2})^{-1}\Sigma^{-1}U^{\mathrm{T}}b^{(j)}$. Let $v^{(j)} = (I + \Sigma^{-2})^{-1}\Sigma^{-1}U^{\mathrm{T}}b^{(j)}$ which gives $x^{*(j)} = Vv^{(j)}$.

Similarly, $\widetilde{x}^{(j)} = V(V^{\mathrm{T}}\boldsymbol{S}_j^{\mathrm{T}}\boldsymbol{S}_jV + \Sigma^{-2})^{-1}\Sigma^{-1}U^{\mathrm{T}}b^{(j)}$. Writing $V^{\mathrm{T}}\boldsymbol{S}_j^{\mathrm{T}}\boldsymbol{S}_jV = I_n + E$, we have

$$\begin{aligned}\widetilde{x}^{(j)} &= V(I + \Sigma^{-2} + E)^{-1}\Sigma^{-1}U^{\mathrm{T}}b^{(j)} \\ &= V(I + (I + \Sigma^{-2})^{-1}E)^{-1}(I + \Sigma^{-2})^{-1}\Sigma^{-1}U^{\mathrm{T}}b^{(j)} \\ &= V(I + (I + \Sigma^{-2})^{-1}E)^{-1}v^{(j)}.\end{aligned}$$

As $\|E\|_2 \le 1/2$, the inverse $(I+(I+\Sigma^{-2})^{-1}E)^{-1}$ is well-defined. Since the matrix V has orthonormal columns, $\|\widetilde{x}^{(j)} - x^{*(j)}\|_2 = \|(I+(I+\Sigma^{-2})^{-1}E)^{-1}v^{(j)} - v^{(j)}\|_2$. Let $(I+(I+\Sigma^{-2})^{-1}E)^{-1}v^{(j)} = v^{(j)} + \Delta$ and we have $v^{(j)} = v^{(j)} + (I+\Sigma^{-2})^{-1}Ev^{(j)} + (I+(I+\Sigma^{-2})^{-1}E)\Delta$ which implies $(I+(I+\Sigma^{-2})^{-1}E)\Delta = -(I+\Sigma^{-2})^{-1}Ev^{(j)}$. Finally,

$$\begin{aligned}(1/2)\|\Delta\|_2 &\le \sigma_{\min}(I+(I+\Sigma^{-2})^{-1}E)\|\Delta\|_2\\ &\le \|(I+(I+\Sigma^{-2})^{-1}E)\Delta\|_2\\ &= \|(I+\Sigma^{-2})^{-1}Ev^{(j)}\|_2 \le \|Ev^{(j)}\|_2,\end{aligned}$$

which gives $\|x^{*(j)} - \widetilde{x}^{(j)}\|_2 = \|V\Delta\|_2 \le 2\|Ev^{(j)}\|_2$. If the matrix $\boldsymbol{S}_j$ has a $\sqrt{\varepsilon/n}$-AMM property i.e.,

$$\begin{aligned}\|V^{\mathrm{T}}\boldsymbol{S}_j^{\mathrm{T}}\boldsymbol{S}_jVv^{(j)} - V^{\mathrm{T}}Vv^{(j)}\|_2 &\le \sqrt{\varepsilon/n}\|V\|_{\mathrm{F}}\|v^{(j)}\|_2\\ &= \sqrt{\varepsilon}\|v^{(j)}\|_2,\end{aligned}$$

we have $\|Ev^{(j)}\|_2 \le \sqrt{\varepsilon}\|Vv^{(j)}\|_2$ and that $\|x^{*(j)} - \widetilde{x}^{(j)}\|_2 \le 2\sqrt{\varepsilon}\|v^{(j)}\|_2 = 2\sqrt{\varepsilon}\|x^{*(j)}\|_2$. □

Proof of Theorem 8.3.1. By a union bound, with probability $\ge 9/10$, in all t iterations, we can assume that the matrices $\boldsymbol{S}_j$ have both the subspace embedding property for the column space of A^{T}, and the AMM property for V and $v^{(j)}$.

From Lemma 8.3.2, $\|\hat{x} - x^*\|_2 = \|\widetilde{x}^{(t)} + \sum_{i=1}^{t-1}\widetilde{x}^{(i)} - x^*\|_2 = \|\widetilde{x}^{(t)} - x^{*(t)}\|_2 \le (\sqrt{\varepsilon})\|x^{*(t)}\|_2$. We also have

$$x^* = x^{*(j-1)} + \sum_{i=1}^{j-2}\widetilde{x}^{(i)} = x^{*(j)} + \sum_{i=1}^{j-1}\widetilde{x}^{(i)}$$

which implies $x^{*(j)} = x^{*(j-1)} - \widetilde{x}^{(j-1)}$ and therefore, $\|x^{*(j)}\|_2 = \|\widetilde{x}^{(j-1)} - x^{*(j-1)}\|_2 \le \sqrt{\varepsilon}\|x^{*(j-1)}\|_2$ for all j, where the last inequality follows from Lemma 8.3.3. Now noting that $x^{*(1)} = x^*$, we obtain $\|\hat{x} - x^*\|_2 \le (\sqrt{\varepsilon})^t\|x^*\|_2$ and using the Pythagorean theorem,

$$\begin{aligned}\mathrm{cost}(\hat{x}) &\le \mathrm{OPT} + (\sigma^2 + \lambda)\|\hat{x} - x^*\|_2^2\\ &\le \mathrm{OPT} + (\sigma^2+\lambda)\varepsilon^t\|x^*\|_2^2.\end{aligned}$$

As $\lambda\|x^*\|_2^2 \le \mathrm{OPT}$, we obtain the result. □

We now show that the OSNAP distribution has both the properties required by Algorithm 8.1.

8.3.1 Properties of OSNAP

Nelson and Nguyên [NN13] proposed OSNAP, an oblivious subspace embedding. OSNAP embeddings are parameterized by their number m of rows and their sparsity s. Essentially, OSNAP is a random $m \times d$ matrix $\boldsymbol{S}$, with each column having exactly s nonzero entries at random locations. Each nonzero entry is $\pm 1/\sqrt{s}$ with probability $1/2$ each. They show that if the positions of the nonzero entries satisfy an "expectation" property and if the nonzero values are drawn from a k-wise independent distribution for a sufficiently large k, then $\boldsymbol{S}$ is an OSE.

Theorem 8.3.4 (Informal, [NN13])**.** *If $m = O(n^{1+\gamma} \operatorname{poly}(\log(n/\varepsilon))/\varepsilon^2)$ and $s = O(1/\gamma\varepsilon)$, then* ***OSNAP*** *is an ε-OSE for n dimensional spaces. Further, $t_{SA} = O(\operatorname{nnz}(A)/\gamma\varepsilon)$ for any $d \times n$ matrix A.*

We show that **OSNAP** with *any* sparsity parameter s and $m = \Omega(1/\varepsilon^2)$ has the ε-AMM property. We state our result as the following lemma.

Lemma 8.3.5. ***OSNAP*** *with $m = \Omega(1/\varepsilon^2\delta)$ and sparsity parameter $s \geq 1$ has the (ε, δ)-AMM property.*

8.3.2 Running times Comparison

As discussed in the introduction, the algorithm of [CYD18] is better than ours when $O(\log(1/\varepsilon))$ passes over the matrix A are allowed, as we require a fresh $1/2$ subspace embedding in each iteration, and they require only one $1/2$ subspace embedding. However, our algorithm is faster when the algorithm is restricted to $t = O(1)$ passes over the input. We compare the running time of our algorithm with theirs when both algorithms are run only for 1 iteration to obtain $1+\varepsilon$ approximate solutions. For ease of exposition, we consider the case when $\sigma^2/\lambda = O(1)$.

From Theorem 8.1.1, the algorithm of [CYD18] requires a $c\sqrt{\varepsilon}$ subspace embedding to output a $1 + \varepsilon$ approximation to ridge regression. By applying a sequence of CountSketch and **OSNAP** sketches, we can obtain a $c\sqrt{\varepsilon}$ embedding with $m = n \operatorname{poly}(\log(n))/\varepsilon$ and $t_{SA^{\mathrm{T}}} = O(\operatorname{nnz}(A) + n^3 \operatorname{poly}(\log(n))/\sqrt{\varepsilon})$ or by directly applying **OSNAP**, we obtain $m = n \operatorname{poly}(\log(n))/\varepsilon$ and $t_{SA^{\mathrm{T}}} = O(\operatorname{nnz}(A) \operatorname{poly}(\log(n))/\sqrt{\varepsilon})$.

From Theorem 8.3.1, our algorithm needs a random matrix that has the $1/2$ subspace embedding property and the $c\sqrt{\varepsilon/n}$-AMM property to compute a $1+\varepsilon$ approximation. **OSNAP** with $m = O(n/\varepsilon + n \operatorname{poly}(\log(n)))$ and $s = O(\operatorname{poly}(\log(n)))$ has this property giving $t_{SA^{\mathrm{T}}} = O(\operatorname{nnz}(A) \operatorname{poly}(\log(n)))$.

Finally, the total time to compute $\widetilde{x}$ is

$$O(t_{SA^{\mathrm{T}}} + mn^{\omega-1} + n^{\omega}),$$

where $\omega < 3$ denotes the matrix multiplication exponent. For the algorithm of [CYD18], depending on the sketching matrices used as described above, the total running time is either

$$O(\operatorname{nnz}(A) + n^3 \operatorname{poly}(\log(n))/\sqrt{\varepsilon} + n^{\omega} \operatorname{poly}(\log(n))/\varepsilon)$$

or

$$O(\operatorname{nnz}(A) \operatorname{poly}(\log(n))/\sqrt{\varepsilon} + n^{\omega} \operatorname{poly}(\log(n))/\varepsilon).$$

For Algorithm 8.1 with $t = 1$, the total running time is $O(\operatorname{nnz}(A) \operatorname{poly}(\log(n)) + n^{\omega} \operatorname{poly}(\log(n))/\varepsilon)$. Thus, we have that when $\operatorname{nnz}(A) \approx n^{\omega}/\varepsilon$, our algorithm is asymptotically faster than their algorithm, as our running time does not have the n^3 term and $\mathrm{nnz}(A)/\sqrt{\varepsilon}$ terms. We note that although the fastest matrix multiplication algorithms are sometimes considered impractical, Strassen's algorithm is already practical for reasonable values of n, and gives $\omega < \log_2 7$. If we consider the algorithm of [CYD18] using just the OSNAP embedding, our algorithm is faster by a factor of $1/\sqrt{\varepsilon}$, which could be substantial when ε is small.

Even non-asymptotically, our result shows that we can replace the sketching matrix in their algorithm with a sketching matrix that is both sparser and has fewer rows, while still obtaining a $1+\varepsilon$ approximation. Both of these properties help the algorithm to run faster.

8.4 Applications to Kernel Ridge Regression

A function $k : X \times X \to \mathbb{R}$ is called a positive semi-definite kernel if it satisfies the following two conditions: (i) For all $x, y \in X$, $k(x,y) = k(y,x)$, and (ii) for any finite set $S = \{ s_1, \dots, s_t \} \subseteq X$, the matrix $K = [k(s_i, s_j)]_{i,j\in[t]}$ is positive semi-definite. Mercer's theorem states that a function $k(\cdot,\cdot)$ is a positive semi-definite kernel as defined above if and only if there exists a function ϕ such that for all $x, y \in X$, $k(x,y) = \phi(x)^{\mathrm{T}}\phi(y)$. Many machine learning algorithms only work with inner products of the data points and therefore all such algorithms can work using the function k directly instead of the explicit mapping ϕ, which in principle could even be infinite dimensional, for example, as in the case of the Gaussian kernel.

Let the rows of a matrix A be the input data points $a_1, \dots, a_n$, and let $\phi(A)$ denote the matrix obtained by applying the function ϕ to each row of the matrix A. The kernel ridge regression problem (see [Mur12] for more details) is defined as

$$c^* = \arg\min_{c} \|\phi(A)\cdot c - b\|_2^2 + \lambda\|c\|_2^2.$$

We have that $c^* = \phi(A)^{\mathrm{T}}(\phi(A)\cdot\phi(A)^{\mathrm{T}} + \lambda I)^{-1}b$ and the value predicted for an input x is given by $\phi(x)^{\mathrm{T}}c^* = \phi(x)^{\mathrm{T}}\phi(A)^{\mathrm{T}}(\phi(A)\cdot\phi(A)^{\mathrm{T}} + \lambda I)^{-1}b$. Letting $\beta = (\phi(A)\phi(A)^{\mathrm{T}} + \lambda I)^{-1}b$ we have $\phi(x)^{\mathrm{T}}c^* = \sum_i k(a_i, x)\beta_i$. Now, note that the (i,j)-th entry of the matrix $K \coloneqq \phi(A)\cdot\phi(A)^{\mathrm{T}}$ is given by $k(a_i, a_j)$ and therefore, to solve the kernel ridge regression problem, we do not need the explicit map $\phi(\cdot)$ and can work directly with the kernel function. Nevertheless, to construct the matrix K, we need to query the kernel function k for $\Theta(n^2)$ pairs of inputs, which may be prohibitive if the kernel evaluation is slow.

Our result for ridge regression shows that if $\boldsymbol{S}$ is a $1/2$ subspace embedding and gives an $\varepsilon/2\sqrt{n}$ AMM guarantee, then

$$\widetilde{c} = \phi(A)^{\mathrm{T}}\cdot(\phi(A)\cdot \boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\cdot\phi(A)^{\mathrm{T}} + \lambda I)^{-1}b$$

satisfies $\|\widetilde{c} - c^*\| \le \varepsilon\|c^*\|_2$ and if $\widetilde{\beta} \coloneqq (\phi^{\mathrm{T}}(A)\cdot \boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\cdot\phi(A)^{\mathrm{T}} + \lambda I)^{-1}b$, then for a new input x, the prediction on x can be computed as $\sum_i k(a_i, x)\widetilde{\beta}_i$. For polynomial kernels, $k(x,y) = \langle x, y\rangle^p$, given the matrix A, it is possible to compute $\boldsymbol{S} \cdot \phi(A)^{\mathrm{T}}$ for a random matrix $\boldsymbol{S}$ that satisfies both the subspace embedding property and the AMM property, and hence obtain $\widetilde{\beta}$ without computing the kernel matrix. We prove the following theorem which follows from Theorems 1 and 3 of [AKK+20].

Theorem 8.4.1. *For all positive integers n, d, p, there exists a distribution on linear sketches $\Pi^p \in \mathbb{R}^{m \times d^p}$ parameterized by sparsity s such that: if $m = \Omega(p/\varepsilon^2)$ and any sparsity s, then Π^p has the ε-AMM property, while if $m = \widetilde{\Omega}(p^4 n/\varepsilon^2)$ and $s = \widetilde{\Omega}((p^4/\varepsilon^2 \operatorname{poly}(\log(nd/\varepsilon)))$, then Π^p has the ε subspace embedding property. Further, given any matrix $A \in \mathbb{R}^{n \times d}$, the matrix $\Pi^p \cdot \phi(A)^{\mathrm{T}}$ for $\phi(x) = x^{\otimes p}$ can be computed in $\widetilde{O}(pnm + ps \cdot \operatorname{nnz}(A))$ time.*

To prove the above theorem, we show that the construction of [AKK+20] gives the above theorem when S_{base} is taken to be TensorSketch and T_{base} is taken to be **OSNAP**. We first prove a lemma which shows that the **OSNAP** distribution has the JL-moment property. For a random variable $\boldsymbol{X}$, let $\|\boldsymbol{X}\|_{L^t} := (\mathbf{E}[|\boldsymbol{X}|^t])^{1/t}$.

Definition 8.4.2 (JL-Moment Property)**.** For every positive integer t and parameters $\varepsilon, \delta \geq 0$, we say a random matrix $\boldsymbol{S} \in \mathbb{R}^{m \times d}$ satisfies the (ε, δ, t)-JL moment property if for any $x \in \mathbb{R}^d$ with $\|x\|_2 = 1$,

$$\| \|\boldsymbol{S}x\|_2^2 - 1\|_{L^t} \leq \varepsilon \delta^{1/t} \text{ and } \mathbf{E}[\|\boldsymbol{S}x\|_2^2] = 1.$$

Lemma 8.4.3. *If $\boldsymbol{S}$ is an **OSNAP** matrix with $m = \Omega(1/\delta\varepsilon^2)$ rows and any sparsity parameter $s \geq 1$, then $\boldsymbol{S}$ has the $(\varepsilon, \delta, 2)$-JL moment property.*

Proof. For $i \in [m]$ and $j \in [d]$, let $\delta_{i,j}$ be the indicator random variable that denotes if the (i, j)-th entry of the matrix $\boldsymbol{S}$ is nonzero. We have that $\sum_i \delta_{i,j} = s$ and that for any $S \subseteq [m] \times [d]$, $\mathbf{E}[\Pi_{(i,j) \in S} \delta_{i,j}] \leq (s/m)^{|S|}$. Also, let $\sigma_{i,j}$ be the sign of the (i, j)-th entry and let $\sigma_{i,j}$ be 4-wise independent Rademacher random variables. Now,

$$\begin{aligned}
\|\boldsymbol{S}x\|_2^2 = \sum_i |\boldsymbol{S}_{i*}x|^2 &= \frac{1}{s} \sum_i (\sum_j \delta_{i,j}\sigma_{i,j}x_j)^2 \\
&= \frac{1}{s} \sum_i \sum_{j,j'} \delta_{i,j}\delta_{i,j'}\sigma_{i,j}\sigma_{i,j'}x_j x_{j'} \\
&= \frac{1}{s} \sum_i \sum_j (\delta_{i,j})^2(\sigma_{i,j})^2 x_j^2 + \frac{1}{s} \sum_i \sum_{j \neq j'} \delta_{i,j}\delta_{i,j'}\sigma_{i,j}\sigma_{i,j'}x_j x_{j'}.
\end{aligned}$$

We have $\delta_{i,j}^2 = \delta_{i,j}$ and $\sigma_{i,j}^2 = 1$ for all i, j. So,

$$\frac{1}{s} \sum_i \sum_j (\delta_{i,j})^2(\sigma_{i,j})^2 x_j^2 = \frac{1}{s} \sum_i \sum_j \delta_{i,j} x_j^2 = \frac{1}{s} \sum_j x_j^2 \sum_i \delta_{i,j} = \frac{1}{s} \sum_j x_j^2 \cdot s = \|x\|_2^2 = 1.$$

Therefore,

$$\|Sx\|_2^2 = 1 + \frac{1}{s} \sum_i \sum_{j \neq j'} \delta_{i,j}\delta_{i,j'}\sigma_{i,j}\sigma_{i,j'}x_j x_{j'}.$$

If $\sigma_{i,j}$ are uniform random signs that are 2-wise independent, then for $j \neq j'$, $\mathbf{E}[\sigma_{i,j}\sigma_{i,j'}] = 0$. As the set of random variables $\delta_{i,j}$ are independent of the random variables $\sigma_{i,j}$, we have $\mathbf{E}[\delta_{i,j}\delta_{i,j'}\sigma_{i,j}\sigma_{i,j'}] = \mathbf{E}[\delta_{i,j}\delta_{i,j'}]\,\mathbf{E}[\sigma_{i,j}\sigma_{i,j'}] = 0$ for $j \neq j'$ which implies that $\mathbf{E}[\|Sx\|_2^2] = 1$. We also have

$$(\|Sx\|_2^2 - 1)^2 = \frac{1}{s^2} \sum_{i,i'} \sum_{\substack{j \neq j' \\ k \neq k'}} \delta_{i,j}\delta_{i,j'}\delta_{i',k}\delta_{i',k'}\sigma_{i,j}\sigma_{i,j'}\sigma_{i',k}\sigma_{i',k'}x_j x_{j'} x_k x_{k'}.$$

If $i \neq i'$, $j \neq j'$, and $k \neq k'$, then the random variables $\sigma_{i,j}$, $\sigma_{i,j'}$, $\sigma_{i',k}$, and $\sigma_{i',k'}$ are distinct and if they are 4-wise independent Rademacher random variables, then $\mathbf{E}[\sigma_{i,j}\sigma_{i,j'}\sigma_{i',k}\sigma_{i',k'}] = 0$ which implies that

$$\mathbf{E}[(\|Sx\|_2^2 - 1)^2] = \frac{1}{s^2} \sum_{i} \sum_{\substack{j \neq j' \\ k \neq k'}} \mathbf{E}[\delta_{i,j}\delta_{i,j'}\delta_{i,k}\delta_{i,k'}]\,\mathbf{E}[\sigma_{i,j}\sigma_{i,j'}\sigma_{i,k}\sigma_{i,k'}]x_j x_{j'} x_k x_{k'}.$$

Again, if all the indices j, j', k, k' are distinct, then by the 4-wise independence of the σ random variables, we obtain that $\mathbf{E}[\sigma_{i,j}\sigma_{i,j'}\sigma_{i,k}\sigma_{i,k'}] = 0$, which leaves only $j = k \neq j' = k'$ and $j = k' \neq j' = k$ as the cases where the expectation can be non-zero. In each of these cases, $\sigma_{i,j}\sigma_{i,j'}\sigma_{i,k}\sigma_{i,k'} = 1$ with probability 1. Therefore,

$$\begin{aligned}\mathbf{E}[(\|Sx\|_2^2 - 1)^2] = \frac{2}{s^2}\sum_i \sum_{j \neq j'} \mathbf{E}[\delta_{i,j}\delta_{i,j'}]x_j^2 x_{j'}^2 \leq \frac{2}{s^2}\frac{s^2}{m^2}\sum_i \sum_{j\neq j'} x_j^2 x_{j'}^2 &\leq \frac{2}{m^2}\sum_i (\sum_j x_j^2)(\sum_{j'} x_{j'}^2) \\ &\leq \frac{2}{m}\end{aligned}$$

which gives that $\|\|Sx\|_2^2 - 1\|_{L^2} = \mathbf{E}[(\|Sx\|_2^2 - 1)^2]^{1/2} \leq \sqrt{2/m}$. Now, for $m = \Omega(1/\varepsilon^2\delta)$, we have $\|\|Sx\|_2^2 - 1\|_{L^2} \leq \varepsilon\delta^{1/2}$, which proves that the matrix S has the $(\varepsilon, \delta, 2)$-JL moment property. □

We can now prove Theorem 8.4.1

Proof of Theorem 8.4.1. Let $q = 2^{\lceil \log_2(p) \rceil}$. The construction of the sketch for polynomial kernels of [AKK+20] uses two distributions of matrices S_{base} and T_{base}. The proof of Theorem 1 of [AKK+20] requires that the distributions S_{base} and T_{base} have the $(\varepsilon/\sqrt{4q+2}, \delta, 2)$-JL moment property. We take S_{base} to be TensorSketch and T_{base} to be **OSNAP**. As Lemma 8.4.3 shows, **OSNAP** with $m = \Omega(q/\delta\varepsilon^2)$ and any sparsity s has the $(\varepsilon/\sqrt{4q+2}, \delta, 2)$-JL moment property.

From Theorem 3 of [AKK+20], we also have that for $m = \widetilde{\Omega}(p^4 n/\varepsilon^2)$ and sparsity parameter $s = \widetilde{\Omega}((p^4/\varepsilon^2) \cdot \text{poly}(\log(nd/\varepsilon)))$, the sketch has the ε-subspace embedding property. The running time of applying the sketch to $\phi(A)^{\mathrm{T}}$ also follows from the same theorem. $\square$

Thus, for the sketch to have both the 1/2 subspace embedding property and the $\varepsilon/\sqrt{4n}$ AMM property, we need to take $m = \widetilde{\Omega}(p^4 n + pn/\varepsilon^2)$ and $s = \widetilde{\Omega}(p^4 \operatorname{poly}(\log(nd)))$. The time to compute $\Pi^p \cdot \phi(A)^{\mathrm{T}}$ is $\widetilde{O}(p^5 \operatorname{nnz}(A) + p^5 n^2 + p^2 n/\varepsilon^2)$ and the time to compute $\widetilde{\beta}$ is $\widetilde{O}(p^5 \operatorname{nnz}(A) + p^5 n^2 + p^2 n^2/\varepsilon^2 + p^4 n^{\omega} + pn^{\omega}/\varepsilon^2)$, thereby obtaining a near-input sparsity time algorithm for polynomial kernel ridge regression.

8.5 Lower bounds

Dimensionality reduction, by multiplying the input matrix A on the right with a random sketching matrix, seems to be the most natural way to speed up ridge regression. Recall that in our algorithm above, we show that we only need the sketching distribution to satisfy a simple AMM guarantee, along with being a constant factor subspace embedding, to be able to obtain a $1+\varepsilon$ approximation. We show that, in this natural framework, the bounds on the number of rows required for a sketching matrix we obtain are nearly optimal for all "non-dilating" distributions.

More formally, we show lower bounds in the restricted setting where for an oblivious random matrix $\boldsymbol{S}$, the vector $\widetilde{x} = A^{\mathrm{T}}(A\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}A^{\mathrm{T}} + \lambda I)^{-1}b$ must be a $1+\varepsilon$ approximation to the ridge regression problem with probability $\geq 99/100$. We show that the matrix $\boldsymbol{S}$ must at least have $m = \Omega(n\sigma^2/\lambda\varepsilon)$ rows if $\boldsymbol{S}$ is "non-dilating".

Definition 8.5.1 (Non-Dilating Distributions)**.** A distribution $\mathcal{S}$ over $m \times d$ matrices is a Non-Dilating distribution if for all $d \times n$ orthonormal matrices V,

$$\mathbf{Pr}_{\boldsymbol{S}\sim\mathcal{S}}[\|\boldsymbol{S}V\|_2 \leq O(1)] \geq 99/100.$$

Most sketching distributions proposed in previous work satisfy the property $\mathbf{E}[V^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}V] = V^{\mathrm{T}}V = I$. Thus, the condition of non-dilation is not very restrictive. For example, a Gaussian distribution with $O(n)$ rows satisfies this condition, and other sketching distributions such as SRHT, CountSketch, and OSNAP with $O(n\log(n))$ rows all satisfy this condition with $O(1)$ replaced by at most $O(\log(n))$. Though we prove our lower bounds for non-dilating distributions with $O(1)$ distortion, the lower bounds also hold with distributions with $O(\log(n))$ distortion with at most an $O(\log(n))$ factor loss in the lower bound.

For $n' \geq n$, let $O^{n'\times n}$ denote the collection of $n' \times n$ orthonormal matrices $V \in \mathbb{R}^{n'\times n}$ i.e., $V^{\mathrm{T}}V = I_n$. Without loss of generality, we assume that $\lambda = 1$.

Assume that there is a distribution $\mathcal{S}$ over $m \times d$ matrices such that given an arbitrary matrix $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ such that for $\boldsymbol{S} \sim \mathcal{S}$, with probability $\geq 99/100$,

$$\|A\widetilde{x} - b\|_2^2 + \|\widetilde{x}\|_2^2 \leq (1+\varepsilon)\mathrm{OPT},$$

where $\widetilde{x} = A^{\mathrm{T}}(AS^{\mathrm{T}}SA^{\mathrm{T}} + I)^{-1}b$. Given an instance (A, b), let $\boldsymbol{S}$ be a good$_{A,b}$ matrix if the above event holds, i.e., $\widetilde{x}$ is a $1 + \varepsilon$ approximation. Let b be a fixed unit vector. Thus, from our assumption,

$$\mathbf{Pr}_{\boldsymbol{U}\sim O^{n\times n}, \boldsymbol{V}\sim O^{d\times n}, \boldsymbol{S}\sim\mathcal{S}}[\boldsymbol{S} \text{ is good}_{\sigma\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}, b}] \geq 99/100. \tag{8.3}$$

For the problem $(\sigma\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}, b)$ where b is a fixed unit vector, we have $\mathsf{OPT} = 1/(1+\sigma^2)$. We also have for $v = (\Sigma^{-2} + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1}\Sigma^{-1}\boldsymbol{U}^{\mathrm{T}}b$ that

$$\begin{aligned}\operatorname{cost}(\widetilde{x}) - \mathsf{OPT} &= v^{\mathrm{T}}E\Sigma(I - (\Sigma^2 + I)^{-1})\Sigma E v \\ &\geq \lambda_{\min}(I - (\Sigma^2 + I)^{-1})\|\Sigma E v\|_2^2,\end{aligned}$$

where $E = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V} - \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}$, which is the error in approximating the identity matrix using the sketch $\boldsymbol{S}$, and Σ is the matrix of singular values of $\sigma\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}$. In our case, $\Sigma = \sigma I_n$ for some $\sigma \geq 1$ which implies that

$$\operatorname{cost}(\widetilde{x}) - \mathsf{OPT} \geq \frac{1}{2}\|E(\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1}\boldsymbol{U}^{\mathrm{T}}b\|_2^2$$

once we cancel out Σ and Σ^{-1}. Thus, if $\boldsymbol{S}$ is good$_{(\sigma\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}, b)}$,

$$\|E(\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1}\boldsymbol{U}^{\mathrm{T}}b\|_2^2 \leq \frac{2\varepsilon}{1+\sigma^2} \leq \frac{2\varepsilon}{\sigma^2}.$$

Therefore, $\mathbf{Pr}_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{S}}[\|E(\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1}\boldsymbol{U}^{\mathrm{T}}b\|_2^2 \leq 2\varepsilon/\sigma^2] \geq \mathbf{Pr}_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{S}}[\boldsymbol{S} \text{ is good}_{\sigma\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}, b}] \geq 99/100$. Now, for a fixed unit vector b, the vector $\boldsymbol{U}^{\mathrm{T}}b$ is a uniformly random unit vector that is independent of $\boldsymbol{V}$ and $\boldsymbol{S}$. Thus,

$$\mathbf{Pr}_{\boldsymbol{V},\boldsymbol{S},\boldsymbol{r}}[\|E(\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1}\boldsymbol{r}\|_2^2 \leq 2\varepsilon/\sigma^2] \geq 0.99,$$

where above and throughout the section, $\boldsymbol{r}$ is a uniformly random unit vector. Now we transform this property of the random matrix $\boldsymbol{S}$ into a probability statement about the Frobenius norm of a certain matrix.

Lemma 8.5.2 (Random vector to Frobenius Norm). *If $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a random matrix independent of the random uniform vector $\boldsymbol{r}$ such that $\mathbf{Pr}_{\boldsymbol{M},\boldsymbol{r}}[\|\boldsymbol{M}\boldsymbol{r}\|_2^2 \leq a] \geq 99/100$, then $\mathbf{Pr}_{\boldsymbol{M}}[\|\boldsymbol{M}\|_{\mathrm{F}}^2 \leq Can] \geq 9/10$ for large enough constant C.*

To prove the lemma, we first prove the following similar lemma in which the matrix M is a deterministic matrix.

Lemma 8.5.3. *Let $M \in \mathbb{R}^{n\times n}$ be a fixed matrix and $\boldsymbol{r}$ be a uniformly random unit vector. If $\mathbf{Pr}_{\boldsymbol{r}}[\|M\boldsymbol{r}\|_2^2 \leq a] \geq 9/10$, then $\|M\|_{\mathrm{F}}^2 \leq Cna$ for a large enough universal constant C.*

Proof. Let $\boldsymbol{g} \in \mathbb{R}^n$ be a Gaussian random vector with i.i.d. entries drawn from $N(0,1)$. Then the distribution of $\boldsymbol{g}/\|\boldsymbol{g}\|_2$ is identical to that of a uniformly random unit vector in n dimensions by rotational invariance of the Gaussian distribution. Therefore, from our assumption, $\mathbf{Pr}_{\boldsymbol{g}}[\|M\boldsymbol{g}\|_2^2 \leq a\|\boldsymbol{g}\|_2^2] \geq 9/10$. We also have that with probability $\geq 9/10$, $\|\boldsymbol{g}\|_2^2 \leq C_1 n$ for a large enough absolute constant C_1. Thus, we have by a union bound that,

$$\mathbf{Pr}_{\boldsymbol{g}}[\|M\boldsymbol{g}\|_2^2 \leq a\|\boldsymbol{g}\|_2^2 \ \wedge\ \|\boldsymbol{g}\|_2^2 \leq C_1 n] \geq 8/10,$$

which implies that

$$\mathbf{Pr}_{\boldsymbol{g}}[\|M\boldsymbol{g}\|_2^2 \leq C_1 an] \geq 8/10.$$

Let $M = U\Sigma V^{\mathrm{T}}$ be the singular value decomposition of the matrix M. Then, the above equation is equivalent to

$$8/10 \leq \mathbf{Pr}_{\boldsymbol{g}}[\|\Sigma V^{\mathrm{T}}\boldsymbol{g}\|_2^2 \leq C_1 an] = \mathbf{Pr}_{\boldsymbol{g}}[\|\Sigma\boldsymbol{g}\|_2^2 \leq C_1 an]$$

where the equality follows from the fact that for an orthonormal matrix V^{T}, we have $V^{\mathrm{T}}\boldsymbol{g} \equiv \boldsymbol{g}$. Thus, if the singular values of M are $\sigma_1, \ldots, \sigma_n$, we have

$$\mathbf{Pr}_{\boldsymbol{g}}[\sum_i \sigma_i^2 \boldsymbol{g}_i^2 \leq C_1 an] \geq 8/10.$$

Now, we have the following lemma which gives an upper bound on the probability of a linear combination of squared Gaussian random variables being small.

Lemma 8.5.4 ([Low12]). *If $a_i \geq 0$ for $i = 1, \ldots, n$ are constants and $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n$ are i.i.d. Gaussian random variables of mean 0 and variance 1, then for every $\delta > 0$,*

$$\mathbf{Pr}[\sum_i a_i \boldsymbol{g}_i^2 \leq \delta \sum_i a_i] \leq e\sqrt{\delta}.$$

Proof. The inequality is obviously true for $\delta \geq 1$. We now consider arbitrary $\delta < 1$. Assume without loss of generality that $\sum_i a_i = 1$. Now, for any $\lambda > 0$,

$$\begin{aligned}\mathbf{Pr}[\sum_i a_i \boldsymbol{g}_i^2 \leq \delta] &= \mathbf{Pr}[-\lambda \sum_i a_i \boldsymbol{g}_i^2 \geq -\lambda\delta] \\ &= \mathbf{Pr}[\exp(-\lambda \sum_i a_i \boldsymbol{g}_i^2) \geq \exp(-\lambda\delta)] \leq \exp(\lambda\delta)\,\mathbf{E}[\exp(-\lambda \sum_i a_i \boldsymbol{g}_i^2)]\end{aligned}$$

and therefore,

$$\begin{aligned}\mathbf{Pr}[\sum_i a_i \boldsymbol{g}_i^2 \leq \delta] &\leq \exp(\lambda\delta)\,\mathbf{E}[\exp(-\lambda \sum_i a_i \boldsymbol{g}_i^2)] \\ &= \exp(\lambda\delta) \prod_i \mathbf{E}[\exp(-\lambda a_i \boldsymbol{g}_i^2)] \\ &= \exp(\lambda\delta) \prod_i (1 + 2\lambda a_i)^{-1/2}.\end{aligned}$$

Now, $\prod_i (1 + 2\lambda a_i) \geq 1 + 2\lambda(\sum_i a_i) = 1 + 2\lambda$ which implies that $\prod (1 + 2\lambda a_i)^{-1/2} \leq (1 + 2\lambda)^{-1/2}$

which gives

$$\mathbf{Pr}[\sum_i a_i \boldsymbol{g}_i^2 \le \delta] \le \exp(\lambda\delta)(1+2\lambda)^{-1/2}.$$

Picking $\lambda \ge 0$ such that $1 + 2\lambda = 1/\delta$, we obtain that $\mathbf{Pr}[\sum_i a_i \boldsymbol{g}_i^2 \le \delta] \le \exp((1-\delta)/2)\sqrt{\delta} \le e\sqrt{\delta}$. □

For $\delta = 0.01$, the above lemma implies that $\mathbf{Pr}[\sum_i \sigma_i^2 \boldsymbol{g}_i^2 \le 0.01 \sum_i \sigma_i^2] \le e \cdot (0.1) \le 0.3$. This, in particular implies that $0.01 \sum_i \sigma_i^2 = 0.01\|\Sigma\|_{\mathrm{F}}^2 \le C_1 an$ which gives $\|M\|_{\mathrm{F}}^2 = \|\Sigma\|_{\mathrm{F}}^2 \le Can$ for a large enough absolute constant C. □

We can now prove the lemma.

Proof of Lemma 8.5.2. Let $\boldsymbol{M}$ be *good* if $\mathbf{Pr}_{\boldsymbol{r}}[\|\boldsymbol{Mr}\|_2^2 \le a] \ge 9/10$ and let $\boldsymbol{M}$ be *bad* otherwise and note from the above lemma that if $\boldsymbol{M}$ is *good*, then $\|\boldsymbol{M}\|_{\mathrm{F}}^2 \le Can$. Now,

$$\begin{aligned}
&99/100 \le \mathbf{Pr}_{\boldsymbol{M},\boldsymbol{r}}[\|\boldsymbol{Mr}\|_2^2 \le a] \\
&\le \mathbf{Pr}_{\boldsymbol{M}}[\boldsymbol{M} \text{ is } \textit{good}] + \mathbf{Pr}_{\boldsymbol{M}}[\boldsymbol{M} \text{ is } \textit{bad}] \cdot (9/10) \\
&= 9/10 + (1/10) \cdot \mathbf{Pr}_{\boldsymbol{M}}[\boldsymbol{M} \text{ is } \textit{good}]
\end{aligned}$$

which implies that $\mathbf{Pr}_{\boldsymbol{M}}[\boldsymbol{M} \text{ is } \textit{good}] \ge 9/10$ and therefore $\mathbf{Pr}_{\boldsymbol{M}}[\|\boldsymbol{M}\|_{\mathrm{F}}^2 \le Can] \ge \mathbf{Pr}_{\boldsymbol{M}}[\boldsymbol{M} \text{ is good}] \ge 9/10$. □

This lemma implies that for any random matrix $\boldsymbol{S}$ satisfying (8.3), we have

$$\|E(\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1}\|_{\mathrm{F}}^2 \le Cn\varepsilon/\sigma^2$$

with probability $\ge 9/10$ over $\boldsymbol{V}, \boldsymbol{S}$. Using the non-dilating property of $\boldsymbol{S}$ and applying a union bound, we now have with probability $\ge 8/10$,

$$\begin{aligned}
\|E\|_{\mathrm{F}}^2 &\le \frac{\|E(\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1}\|_{\mathrm{F}}^2}{\sigma_{\min}((\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1})^2} \\
&= \frac{Cn\varepsilon/\sigma^2}{\sigma_{\min}((\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V})^{-1})^2} \le O(n\varepsilon/\sigma^2)
\end{aligned}$$

where we used the fact that for any invertible matrix A, $1/\sigma_{\min}(A^{-1}) = \sigma_{\max}(A)$ and $\sigma_{\max}(\sigma^{-2}I + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V}) \le (1/\sigma^2) + \|\boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V}\|_2 = O(1)$ with probability $\ge 9/10$. Thus, a lower bound on the number of rows in the matrix $\boldsymbol{S}$ to obtain, with probability $\ge 8/10$,

$$\|\boldsymbol{V}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{V} - I\|_{\mathrm{F}} \le O(\sqrt{n\varepsilon/\sigma^2}) = O(\sqrt{\varepsilon/n\sigma^2})n \tag{8.4}$$

implies a lower bound on the number of rows of a random matrix $\boldsymbol{S}$ that satisfies (8.3).

8.5.1 Lower bounds for AMM

Lemma 8.5.5. *Given parameters n and error parameter $\varepsilon \le c/\sqrt{n}$ for a small enough constant c, for all $d \ge Cn/\varepsilon^2$, if a random matrix $\boldsymbol{S} \in \mathbb{R}^{m\times d}$ for all matrices $A \in \mathbb{R}^{d\times n}$ satisfies, $\|A^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}A - A^{\mathrm{T}}A\|_{\mathrm{F}} \le \varepsilon\|A^{\mathrm{T}}\|_{\mathrm{F}}\|A\|_{\mathrm{F}}$ with probability $\ge 9/10$, then $m = \Omega(1/\varepsilon^2)$.*

Moreover, the lower bound of $\Omega(1/\varepsilon^2)$ holds even for the sketching matrices that give the following guarantee: $\mathbf{Pr}_{\boldsymbol{A},\boldsymbol{S}}[\|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{A} - I\|_{\mathrm{F}} \le \varepsilon n] \ge 0.9$, where $\boldsymbol{A}$ is a uniformly random $d \times n$ orthonormal matrix independent of the sketch $\boldsymbol{S}$.

Proof. We assume that such a distribution exists with $m \le c/\varepsilon^2$ for a small enough constant c. Let $\boldsymbol{A} \in \mathbb{R}^{d\times n}$ be a uniformly random orthonormal matrix ($\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = I_n$) independent of the sketching matrix. Then we have

$$\mathbf{Pr}_{\boldsymbol{A},\boldsymbol{S}}[\|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{A} - I\|_{\mathrm{F}} \le \varepsilon n] \ge 0.9,$$

as $\|A^{\mathrm{T}}\|_{\mathrm{F}} = \sqrt{n}$. Let $\boldsymbol{S} = U\Sigma V^{\mathrm{T}}$ be the singular value decomposition with $U \in \mathbb{R}^{m\times m}$, $\Sigma \in \mathbb{R}^{m\times m}$ and $V^{\mathrm{T}} \in \mathbb{R}^{m\times d}$. Note that if $\boldsymbol{S}$ is a random matrix that satisfies the AMM property, then $\boldsymbol{S}\cdot\boldsymbol{Q}$ is also a random matrix that satisfies the AMM property where $\boldsymbol{Q}$ is an independent uniformly random orthonormal matrix. Therefore, we can without loss of generality assume that Σ is independent of V^{T} and that V^{T} is a uniformly random orthonormal matrix. Thus, the above condition implies that

$$\mathbf{Pr}_{\boldsymbol{A},V,\Sigma}[\|\boldsymbol{A}^{\mathrm{T}}V\Sigma^2V^{\mathrm{T}}\boldsymbol{A} - I\|_{\mathrm{F}} \le \varepsilon n] \ge 0.9.$$

Using the following lemma, we effectively show that the matrix $V^{\mathrm{T}}\boldsymbol{A}$ in the above statement can be replaced with $(1/\sqrt{d})\hat{\boldsymbol{G}}$, where $\hat{\boldsymbol{G}}$ is a Gaussian matrix of the same dimensions as $V^{\mathrm{T}}\boldsymbol{A}$.

Lemma 8.5.6 (Lemma 3 of [LW21]). *Let $\boldsymbol{G} \sim \mathcal{G}_{d,d}$ and $\boldsymbol{Z} \sim O^{d\times d}$. Suppose that $p, q \le d$ and $\hat{\boldsymbol{G}}$ is the top left $p \times q$ block of $\boldsymbol{G}$ and $\hat{Z}$ is the top left $p \times q$ block of Z. Then $d_{KL}(\frac{1}{\sqrt{d}}\hat{\boldsymbol{G}}\|\hat{Z}) \le C\frac{pq}{d}$, where C is a universal constant. By applying Pinsker's inequality, we obtain that*

$$d_{TV}(\frac{1}{\sqrt{d}}\hat{\boldsymbol{G}}\|\hat{Z}) \le \sqrt{(1/2)d_{KL}(\frac{1}{\sqrt{d}}\hat{\boldsymbol{G}}\|\hat{Z})} \le \sqrt{Cpq/2d}.$$

Now both the matrices V, $\boldsymbol{A}$ can be taken to be the first m and n columns of independent uniform random orthogonal matrices $\boldsymbol{V}'$ and $\boldsymbol{A}'$, respectively. By the properties of the Haar Measure, we obtain that $(\boldsymbol{V}')^{\mathrm{T}}\boldsymbol{A}'$ is also a uniform random orthogonal matrix. Thus, the matrix $V^{\mathrm{T}}\boldsymbol{A}$ can be seen as the top left $m \times n$ sub-matrix of a uniformly random orthogonal matrix. If $nm \le d/100C$, which can be assumed as $m \le c/\varepsilon^2$ for a small enough constant c, we have from the above lemma that

$d_{\mathrm{TV}}(\frac{1}{\sqrt{d}}\boldsymbol{G} \| V^{\mathrm{T}}\boldsymbol{A}) \leq 0.1$ which implies that

$$|\,\mathbf{Pr}_{G,\Sigma}[\|(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I\|_{\mathrm{F}} \leq \varepsilon n] - \mathbf{Pr}_{\boldsymbol{A},V,\Sigma}[\|\boldsymbol{A}^{\mathrm{T}}V\Sigma^2V^{\mathrm{T}}\boldsymbol{A} - I\|_{\mathrm{F}} \leq \varepsilon n]\,| \leq 0.1$$

and therefore

$$\mathbf{Pr}_{G,\Sigma}[\|(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I\|_{\mathrm{F}} \leq \varepsilon n] \geq 0.8, \tag{8.5}$$

where $\boldsymbol{G}$ is an $m \times n$ matrix of i.i.d. normal random variables. We will now show that if $m \ll 1/\varepsilon^2$, then no distribution over matrices Σ satisfies the above condition. Note that $\boldsymbol{G}$ and Σ are independent. We prove this by showing that a random matrix Σ satisfying the above probability statement must satisfy two properties simultaneously that cannot be satisfied unless $m \geq c/\varepsilon^2$ for a large enough constant c.

Let $\boldsymbol{G}_l$ denote the left half of the matrix $\boldsymbol{G}$ and $\boldsymbol{G}_r$ denote the right half of the matrix $\boldsymbol{G}$. We have

$$\|(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I\|_{\mathrm{F}}^2 \geq \frac{1}{d^2}\|\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2\boldsymbol{G}_l\|_{\mathrm{F}}^2$$

which is obtained by considering the Frobenius norm of the bottom-left block of $(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I$. We first have the following lemma.

Lemma 8.5.7. *Let M be a fixed matrix and $\boldsymbol{G}$ be a Gaussian matrix with t columns. Then with probability ≥ 0.9, $\|M\boldsymbol{G}\|_{\mathrm{F}}^2 \geq 0.001t\|M\|_{\mathrm{F}}^2$.*

Proof. Let $M = U\Sigma V^{\mathrm{T}}$. We have $M\boldsymbol{G} = U\Sigma V^{\mathrm{T}}\boldsymbol{G} = U\Sigma\boldsymbol{G}'$ where $\boldsymbol{G}'$ is a Gaussian matrix with t columns. Now, $\|M\boldsymbol{G}\|_{\mathrm{F}}^2 = \|U\Sigma\boldsymbol{G}'\|_{\mathrm{F}}^2 = \|\Sigma\boldsymbol{G}'\|_{\mathrm{F}}^2 = \sum_i\sum_j \sigma_i^2 g_{ij}^2$. By Lemma 8.5.4, $\sum_i\sum_j \sigma_i^2 g_{ij}^2 \geq (\sum_i\sum_j \sigma_i^2) \cdot 0.001$ with probability ≥ 0.9. Now, using the equality $\sum_i\sum_j \sigma_i^2 = \sum_i t\sigma_i^2 = t\|\Sigma\|_{\mathrm{F}}^2 = t\|M\|_{\mathrm{F}}^2$, we finish the proof. □

Thus, conditioned on the matrix $\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2$, we have that with probability ≥ 0.9,

$$\|\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2\boldsymbol{G}_l\|_{\mathrm{F}}^2 \geq 0.001(n/2)\|\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2\|_{\mathrm{F}}^2.$$

Applying the same lemma again, we have with probability ≥ 0.9, $\|\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2\|_{\mathrm{F}}^2 \geq 0.001(n/2)\|\Sigma^2\|_{\mathrm{F}}^2$. Thus, overall with probability ≥ 0.8 over $\boldsymbol{G}$, we have for any fixed Σ that, $\|\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2\boldsymbol{G}_l\|_{\mathrm{F}}^2 \geq \Omega(n^2\|\Sigma^2\|_{\mathrm{F}}^2)$, and therefore,

$$\mathbf{Pr}_{G,\Sigma}[\|\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2\boldsymbol{G}_l\|_{\mathrm{F}}^2 \geq \Omega(n^2\|\Sigma^2\|_{\mathrm{F}}^2)] \geq 0.8.$$

Using a union bound with (8.5), we obtain that with probability ≥ 0.6, it is simultaneously true that

$$\varepsilon^2 n^2 \geq \|(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I\|_{\mathrm{F}}^2 \geq \frac{1}{d^2}\|\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2\boldsymbol{G}_l\|_{\mathrm{F}}^2$$

and

$$\|\boldsymbol{G}_r^{\mathrm{T}}\Sigma^2\boldsymbol{G}_l\|_{\mathrm{F}}^2 \geq \Omega(n^2\|\Sigma^2\|_{\mathrm{F}}^2)$$

which implies that with probability ≥ 0.6, $(1/d^2)\|\Sigma^2\|_{\mathrm{F}}^2 = O(\varepsilon^2)$ i.e., $(1/d^2)\sum_{i=1}^m \sigma_i^4 = O(\varepsilon^2)$. Thus, if $\boldsymbol{S}$ is a random matrix that satisfies the AMM property and if $\sigma_1, \ldots, \sigma_r$ are its singular values, then with probability ≥ 0.6,

$$\sum_i \sigma_i^4 \leq C_1 d^2 \varepsilon^2 \tag{8.6}$$

for a universal constant C_1.

We now obtain a different probability statement about the singular values of the sketching matrix $\boldsymbol{S}$ by considering the sum of squares of the diagonal entries of the matrix $(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I = (1/d)\sum_{i=1}^m \sigma_i^2 \boldsymbol{g}_i\boldsymbol{g}_i^{\mathrm{T}} - I$ where $\boldsymbol{g}_i$ are n dimensional Gaussian vectors. Note that $((1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I)_{jj} = (1/d)\sum_{i=1}^m \sigma_i^2 \boldsymbol{g}_{ij}^2 - 1$. Fix the matrix Σ. Clearly,

$$\|(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I\|_{\mathrm{F}}^2 \geq \sum_{j=1}^n ((1/d)\sum_{i=1}^m \sigma_i^2 \boldsymbol{g}_{ij}^2 - 1)^2.$$

If $\sum_{i=1}^m \sigma_i^2 \leq d/100$, we have that with probability at least 0.9,

$$(1/d)\sum_{i=1}^m \sigma_i^2 \boldsymbol{g}_{ij}^2 \leq (10/d)\,\mathbf{E}[\sum_{i=1}^m \sigma_i^2 \boldsymbol{g}_{ij}^2] \leq (10/d)\cdot(d/100)$$

which implies that $((1/d)\sum_{i=1}^m \sigma_i^2 \boldsymbol{g}_{ij}^2 - 1)^2 \geq 1/4$ with probability ≥ 0.9. Let $j \in [n]$ be *large* if the previous event holds. By a Chernoff bound, with probability ≥ 0.9, there are $\geq n/C_2$ *large* values $j \in [n]$ for a large enough absolute constant C_2. Thus, $\sum_{i=1}^m \sigma_i^2 \leq d/100$ implies that with probability ≥ 0.9, $\|(1/d)G^{\mathrm{T}}\Sigma^2 G - I\|_{\mathrm{F}}^2 \geq (n/C_2)(1/4) = n/4C_2 \geq \varepsilon^2 n^2$ as we assumed that $\varepsilon \leq c/\sqrt{n}$ for a small enough constant c. Now, if $\mathbf{Pr}_\Sigma[\sum_{i=1}^m \sigma_i^2 \leq d/100] > 0.3$, then by the above property for a fixed Σ, $\mathbf{Pr}_{\Sigma,G}[\|(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I\|_{\mathrm{F}}^2 \geq \varepsilon^2 n^2] > 0.2$ which implies that

$$\mathbf{Pr}_{\Sigma,G}[\|(1/d)\boldsymbol{G}^{\mathrm{T}}\Sigma^2\boldsymbol{G} - I\|_{\mathrm{F}}^2 \leq \varepsilon^2 n^2] < 0.8$$

which is a contradiction to (8.5). Thus, $\mathbf{Pr}_\Sigma[\|\Sigma\|_{\mathrm{F}}^2 \leq d/100] < 0.3$ which implies

$$\mathbf{Pr}_\Sigma[\sum_{i=1}^m \sigma_i^2 \geq d/100] \geq 0.7. \tag{8.7}$$

By a union bound on (8.6) and (8.7), with probability ≥ 0.3, it simultaneously holds that

$$\sum_{i=1}^m \sigma_i^4 \leq C_1 d^2 \varepsilon^2 \text{ and } \sum_{i=1}^m \sigma_i^2 \geq d/100.$$

Now,

$$d^2/100^2 \leq \left(\sum_{i=1}^{m} \sigma_i^2\right)^2 \leq m \sum_{i=1}^{m} \sigma_i^4 \leq C_1 m d^2 \varepsilon^2.$$

Here we used the Cauchy-Schwarz inequality which finally implies that $m = \Omega(1/\varepsilon^2)$. Thus, any oblivious distribution that gives AMM with $\varepsilon < c/\sqrt{n}$ for a small enough constant c must have $\Omega(1/\varepsilon^2)$ rows. □

Although the above lemma only shows that an AMM sketch requires $m = \Omega(1/\varepsilon^2)$ for $d \geq Cn/\varepsilon^2$, we can extend it to show the lower bound for $d \geq C/\varepsilon^2$ for a large enough constant C. Note that $C/\varepsilon^2 = \Omega(n)$ since $\varepsilon \leq c/\sqrt{n}$.

Theorem 8.5.8. *Given $n \geq 0$ and $\varepsilon < c/\sqrt{n}$ for a small enough constant c, there are universal constants C, D such that for all $d \geq D/\varepsilon^2$, any distribution that has the ε AMM property for $d \times n$ matrices must have $\geq C/\varepsilon^2$ rows.*

Before proving Theorem 8.5.8, we first prove the following lemma that shows CountSketch preserves the Frobenius norm of a matrix.

Lemma 8.5.9 (CountSketch Preserves Frobenius Norms). *If S is a CountSketch matrix with $m \geq 200/\varepsilon^2$, then for any arbitrary matrix A, with probability $\geq 9/10$,*

$$\|SA\|_F^2 = (1 \pm \varepsilon)\|A\|_F^2.$$

Proof. For any vector x, we have $\mathbf{E}[(\|Sx\|_2^2 - \|x\|_2^2)^2] \leq (2/m)\|x\|_2^4$ if S is a CountSketch matrix with m rows. Now,

$$\mathbf{E}[|\|Sx\|_2^2 - \|x\|_2^2|]^2 \leq \mathbf{E}[(\|Sx\|_2^2 - \|x\|_2^2)^2] \leq (2/m)\|x\|_2^4$$

which implies that $\mathbf{E}[|\|Sx\|_2^2 - \|x\|_2^2|] \leq \sqrt{2/m}\|x\|_2^2$. For any arbitrary matrix A, we have $|\|SA\|_2^2 - \|A\|_2^2| = |\sum_i \|SA_{*i}\|_F^2 - \|A_{*i}\|_F^2| \leq \sum_i |\|SA_{*i}\|_2^2 - \|A_{*i}\|_2^2|$. Thus,

$$\mathbf{E}[|\|SA\|_F^2 - \|A\|_F^2|] \leq \sum_i \mathbf{E}[|\|SA_{*i}\|_2^2 - \|A_{*i}\|_2^2|] \leq \sqrt{2/m} \sum_i \|A_{*i}\|_2^2 = \sqrt{2/m}\|A\|_F^2.$$

For $m \geq 200/\varepsilon^2$, we have $\mathbf{E}[|\|SA\|_F^2 - \|A\|_F^2|] \leq (\varepsilon/10)\|A\|_F^2$. By Markov's inequality, with probability $\geq 9/10$, $\|SA\|_F^2 = (1 \pm \varepsilon)\|A\|_F^2$. □

Proof of Theorem 8.5.8. Given n and $\varepsilon \leq c/\sqrt{n}$ for a small enough constant, assume that for $d = C_1/\varepsilon^2$ for a large enough universal constant C_1, there is a random matrix S with $r < C_2/\varepsilon^2$ rows such that for any fixed matrix $A \in \mathbb{R}^{d \times n}$, with probability $\geq 99/100$,

$$\|A^{\mathrm{T}} \boldsymbol{S}^{\mathrm{T}} \boldsymbol{S} A - A^{\mathrm{T}} A\|_{\mathrm{F}} \leq (\varepsilon/3)\|A^{\mathrm{T}}\|_{\mathrm{F}}\|A\|_{\mathrm{F}}.$$

Now consider an arbitrary matrix $B \in \mathbb{R}^{d' \times n}$ for $d' \geq Cn/\varepsilon^2$ for which the previous lemma applies. Let $\boldsymbol{S}_1$ be a CountSketch matrix with K/ε^2 rows for a large enough K. With probability $\geq 95/100$, we simultaneously have (i) $\|B^{\mathrm{T}}\boldsymbol{S}_1^{\mathrm{T}}\boldsymbol{S}_1 B - B^{\mathrm{T}}B\|_{\mathrm{F}} \leq (\varepsilon/3)\|B^{\mathrm{T}}\|_{\mathrm{F}}\|B\|_{\mathrm{F}} = (\varepsilon/3)\|B\|_{\mathrm{F}}^2$, and (ii) $\|\boldsymbol{S}_1 B\|_{\mathrm{F}} = (1 \pm \varepsilon/3)\|B\|_{\mathrm{F}}$. By picking C_1 large enough, we have that $C_1 \geq K$. Thus, by our assumption, the random matrix $\boldsymbol{S}$ gives the AMM property for the matrix $\boldsymbol{S}_1 A$. Conditioning on the above events, we have with probability $\geq 99/100$ that

$$\begin{aligned}
&\|B^{\mathrm{T}}\boldsymbol{S}_1^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{S}_1 B - B^{\mathrm{T}}\boldsymbol{S}_1^{\mathrm{T}}\boldsymbol{S}_1 B\|_{\mathrm{F}} \\
&\leq (\varepsilon/3)\|B^{\mathrm{T}}\boldsymbol{S}_1^{\mathrm{T}}\|_{\mathrm{F}}\|\boldsymbol{S}_1 B\|_{\mathrm{F}} \\
&\leq (\varepsilon/3)(1+\varepsilon/3)^2\|B\|_{\mathrm{F}}^2 \leq (\varepsilon/2)\|B\|_{\mathrm{F}}^2.
\end{aligned}$$

By the triangle inequality, we obtain $\|B^{\mathrm{T}}\boldsymbol{S}_1^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{S}_1 B - B^{\mathrm{T}}B\|_{\mathrm{F}} \leq (\varepsilon/3 + \varepsilon/2)\|B\|_{\mathrm{F}}^2 \leq \varepsilon\|B\|_{\mathrm{F}}^2$. Thus, by a union bound, with probability ≥ 0.9, the random matrix $\boldsymbol{S} \cdot \boldsymbol{S}_1$ satisfies that for any fixed matrix B with $\geq Cn/\varepsilon^2$ rows, with probability ≥ 0.9,

$$\|B^{\mathrm{T}}(\boldsymbol{S} \cdot \boldsymbol{S}_1)^{\mathrm{T}}(\boldsymbol{S} \cdot \boldsymbol{S}_1)B - B^{\mathrm{T}}B\|_{\mathrm{F}} \leq \varepsilon\|B\|_{\mathrm{F}}^2$$

implying that even for a matrix with at least Cn/ε^2 rows, there is an oblivious sketching distribution with $r < C_2/\varepsilon^2$ rows which gives an AMM guarantee. This contradicts the previous lemma and hence our assumption that there is a small sketching matrix for matrices with $d = C_1/\varepsilon^2$ rows is false. Thus, we have that for any $d \geq C/\varepsilon^2$ for a large enough constant C, there is no sketching distribution with $< C_1/\varepsilon^2$ rows that gives the ε AMM guarantee for matrices with $\geq d$ rows. □

As discussed in the introduction, in proving the above results, we crucially use the fact that a sub-matrix of a random orthonormal matrix is close to a Gaussian matrix in total variation distance to prove the above theorem. This seems to be a useful direction to obtain lower bounds for other sketching problems.

8.5.2 Lower Bound Wrap up

In the case of ridge regression with $\lambda = 1$, (8.4) shows that the sketching distribution has to satisfy the AMM guarantee with parameter $c\sqrt{\varepsilon/n\sigma^2}$. By using the above hardness result for AMM, we obtain the following theorem.

Theorem 8.5.10. *If $\mathcal{S}$ is a non-dilating distribution over $m \times d$ matrices such that for all ridge regression* *instances (A, b, λ) with $A \in \mathbb{R}^{n\times d}$, $1 \le \sigma^2/\lambda \le \alpha$ satisfies,*

$$\mathbf{Pr}_{S\sim\mathcal{S}}[\|A\widetilde{x} - b\|_2^2 + \lambda\|\widetilde{x}\|_2^2 \le (1+\varepsilon)\mathrm{OPT}] \ge 0.99,$$

for $\widetilde{x} = A^{\mathrm{T}}(AS^{\mathrm{T}}SA^{\mathrm{T}} + \lambda I)^{-1}b$, then $m = \Omega(n\alpha/\varepsilon) = \Omega(n\sigma^2/\lambda\varepsilon)$.

8.6 An Experiment

We run our algorithm on a ridge regression instance with a 6000×70000 matrix A whose entries are independent Gaussian random variables. We set λ such that $\sigma^2/\lambda \approx 1$. Naïvely computing $x^* = A^{\mathrm{T}}(AA^{\mathrm{T}} + \lambda I)^{-1}b$ takes $t_{\text{naive}} = 71$ seconds on our machine. We use OSNAP with sparsity $s = 8$ and vary the number r of rows and observe the running times and quality of the solution that is obtained by our algorithm.

Our experiments show the general trends we expect. Increasing the number of rows in the sketching matrix results in a solution $\widetilde{x}$ that has lower cost and also is closer to the optimum solution x^*. We also see that the running time of the algorithm is nearly linear in the sketch size r, implying that the time required to apply the sketch is negligible for this instance. At sketch size $r = 30000$, that is less than $d/2$, we see that the algorithm runs nearly 40% faster than the naïve algorithm while computing a solution that has a cost within 5% of the optimum. For larger values of d, we expect to obtain a greater speedup as compared to naïvely computing x^*.

Notice that we do not compare with the algorithm of [CYD18] as for one iteration, our algorithm is exactly the same as theirs. Our theorems show that the sketch can be smaller and sparser than what is shown in their work to compute $1+\varepsilon$ approximate solutions, giving the first proof of correctness about the quality of the solution at smaller sketch sizes.

8.7 Conclusions and Open Questions

In this work, we relax the requirements in earlier algorithms and show that just a constant subspace embedding paired with a weaker approximate matrix multiplication guarantee suffices to obtain accurate solutions for the ridge regression problem. We also obtain tight lower bounds for oblivious sketches that have ε-AMM property and using it we show that for a specific type of algorithms, the sketching dimension we achieve is essentially optimal.

An interesting question is if the sketching lower bound for ridge regression can be expanded to more general algorithms. Concretely, let λ and σ be given parameters. Suppose that $\boldsymbol{S}$ is a sketching matrix paired with a function f. If for all $n \times d$ matrices A with $\|A\|_2 = \sigma$ and label vectors b, if $x = A^{\mathrm{T}}y$, where $y = f(AS, b)$ satisfies

$$\|Ax - b\|_2^2 + \lambda\|x\|_2^2 \le (1+\varepsilon)\cdot \mathrm{OPT}$$

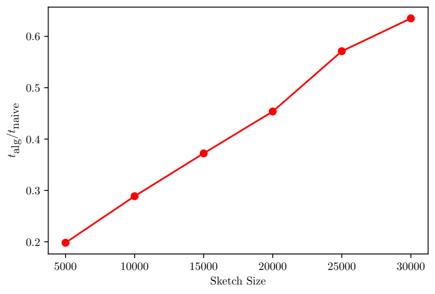


Figure 8.1: $t_{\text{alg}}/t_{\text{naive}}$ vs # of rows of OS-NAP

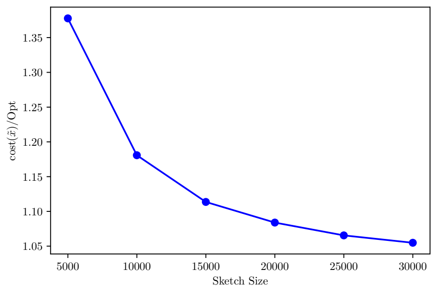


Figure 8.2: $\text{cost}(\widetilde{x})/\text{OPT}$ vs # of rows of OSNAP

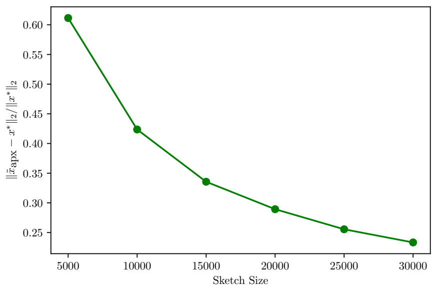


Figure 8.3: $\|\widetilde{x} - x^*\|_2/\|x^*\|_2$ vs # of rows of OSNAP

with probability $\geq 9/10$, can we show that the matrix S requires $\Omega(n\sigma^2/\lambda\varepsilon)$ columns?

Chapter 9

PolySketchFormer: Fast Transformers via Sketching Polynomial Kernels

9.1 Introduction

Transformer-based models [VSP+17] are state-of-the-art for many natural language tasks, leading to breakthroughs in machine translation, language understanding [DCLT19], and language modeling [BMR+20, CND+22, Ope23, ADF+23]. However, the quadratic time and space complexity of the attention mechanism limits scalability for long context lengths. Numerous "efficient transformers" have been proposed to address this issue [WLK+20, KVPF20, CLD+20, HJK+23]. These variants approximate[1] the standard attention mechanism. A survey by Tay, Dehghani, Bahri and Metzler [TDBM22] provides a broad overview of these techniques. While many efficient transformer constructions achieve linear theoretical training complexity, the survey observes that practical training speedups are often less significant, with potential losses in model quality. This explains the continued dominance of vanilla transformers.

In this work, we focus on improving training latency for transformer models in decoding-only tasks, specifically language modeling trained via next-word prediction. We will first briefly discuss existing approaches to make training of transformer models faster and then place our contributions in context.

Memory efficient and I/O aware approach. FlashAttention and FlashAttention-2 [DFE+22, Dao23] seeks to enable vanilla transformer training on long contexts. This is achieved through I/O-aware optimizations like blocking/tiling and rematerialization, significantly improving memory efficiency. While this reduces the $O(n^2)$[2] HBM (High-Bandwidth Memory) requirements of ML accelerators (GPUs/TPUs), enabling fast training on thousands of tokens, the computational cost per training

[1]"Approximation" is used informally here, since some "efficient transformers" deviate significantly from the vanilla model.

[2]n denotes the context length – the number of input tokens.

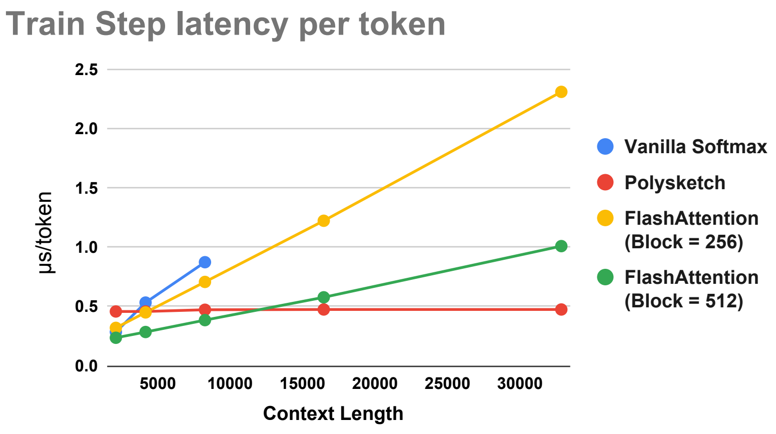


Figure 9.1: Train step latency per token in µs/token of GPT-2 small style models with softmax attention (FlashAttention) v.s. ours. Each model is trained with 1M tokens batches. Vanilla softmax attention goes out-of-memory (OOM) for context lengths > 8k.

step remains $O(n^2)$ (see Figure 9.1), and this remains a barrier to further scaling the context length.

Approximate softmax attention via sparsification. Another line of work tries to approximate softmax attention and avoid $n \times n$ attention matrix computation by focusing on a smaller set of pairs of *query* and *key* vectors. Techniques include utilizing locality/positional information [CGRS19, BPC20, XTC+23, ZGD+20, RSVG21, DMD+23], hashing/bucketing [KKL20, SYY21, HJK+23], low-rank projection [WLK+20], or other sparsification methods. In these cases, there is usually some trade-off between model quality and sparsity, i.e., denser attentions improve quality but decrease speed. Hence, an efficient high-quality $n \times n$ attention mechanism may potentially improve on these techniques.

Efficient $n \times n$ attention by kernel-based methods. The kernel-based view of attention was taken by a series of earlier works [TBY+19, KVPF20, CLD+20, PPY+21]. In particular, let $\{\boldsymbol{q}_i \in \mathbb{R}^h\}_{i\in[n]}$, $\{\boldsymbol{k}_i \in \mathbb{R}^h\}_{i\in[n]}$ and $\{\boldsymbol{v}_i \in \mathbb{R}^h\}_{i\in[n]}$ be sets of *query*, *key* and *value* vectors respectively, the output of the attention mechanism for query $\boldsymbol{q}_i$ is computed as

$$\mathsf{Attn}(\boldsymbol{q}_i, \{\boldsymbol{k}_j\}, \{\boldsymbol{v}_j\}) = \sum_{j\in[n]} \frac{\sigma(\boldsymbol{q}_i, \boldsymbol{k}_j)}{\sum_{j'\in[n]} \sigma(\boldsymbol{q}_i, \boldsymbol{k}_{j'})} \cdot \boldsymbol{v}_j^{\mathrm{T}}.$$

When the similarity kernel function $\sigma(\boldsymbol{x}, \boldsymbol{y}) \coloneqq \exp(\langle \boldsymbol{x}, \boldsymbol{y}\rangle)$, the above attention is exactly the softmax attention[3]. If there exists a feature map ϕ such that $\sigma(\boldsymbol{x}, \boldsymbol{y}) \equiv \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y})\rangle$, the attention

[3]In standard softmax attention, $\sigma(\boldsymbol{x}, \boldsymbol{y}) \coloneqq \exp(\langle \boldsymbol{x}, \boldsymbol{y}\rangle / \sqrt{h})$. We omit $\sqrt{h}$ here for simplicity of the presentation.

output for query $\boldsymbol{q}_i$ can be rewritten as:

$$\mathsf{Attn}(\boldsymbol{q}_i, \{\boldsymbol{k}_j\}, \{\boldsymbol{v}_j\}) = \sum_{j\in[n]} \frac{\phi(\boldsymbol{q}_i)^{\mathrm{T}} \cdot \phi(\boldsymbol{k}_j)}{\sum_{j'\in[n]} \phi(\boldsymbol{q}_i)^{\mathrm{T}} \cdot \phi(\boldsymbol{k}_{j'})} \cdot \boldsymbol{v}_j^{\mathrm{T}}$$
$$= \frac{\phi(\boldsymbol{q}_i)^{\mathrm{T}} \cdot \sum_{j\in[n]} \phi(\boldsymbol{k}_j) \cdot \boldsymbol{v}_j^{\mathrm{T}}}{\phi(\boldsymbol{q}_i)^{\mathrm{T}} \cdot \sum_{j'\in[n]} \phi(\boldsymbol{k}_{j'})}.$$

If $\phi(\cdot)$ has a finite dimension h', one can first compute $\sum_{j'\in[n]} \phi(\boldsymbol{k}_{j'})$ and $\sum_{j\in[n]} \phi(\boldsymbol{k}_j) \cdot \boldsymbol{v}_j^{\mathrm{T}}$ in $O(nhh')$ time, and then compute $\mathsf{Attn}(\boldsymbol{q}_i, \{\boldsymbol{k}_j\}, \{\boldsymbol{v}_j\})$ for all $i \in [n]$ in another $O(nhh')$ time, which is linear in the context length n.

Most of the existing works such as [KVPF20, BFD+22, TBY+19, BMD+23, YWS+23, KPZ+21] only consider similarity functions $\sigma(\boldsymbol{x}, \boldsymbol{y})$ with a low dimensional feature mapping (e.g., $\sigma(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle^2$, $\langle \mathrm{elu}(\boldsymbol{x}) + \mathbf{1}, \mathrm{elu}(\boldsymbol{y}) + \mathbf{1} \rangle$, etc.). Hua, Dai, Liu and Li [HDLL22] proposed to use a mixed strategy based on the positions of the tokens: if positions $i, j \in [n]$ are close enough, they use $\sigma(\boldsymbol{q}_i, \boldsymbol{k}_j) = \mathrm{relu}^2(\langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle)$. Otherwise, they use $\sigma(\boldsymbol{q}_i, \boldsymbol{k}_j) = \langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle$, which again has a low dimensional feature mapping. These simple similarity kernel functions $\sigma(\cdot)$ either suffer from some loss of model quality [KVPF20] or require additional tweaks of network structures (e.g., significantly increasing the number of attention layers [HDLL22], introducing decay factors for earlier tokens [YWS+23]) to achieve comparable model quality as softmax attention.

Some other previous works try to approximate the regular softmax attention via approximate feature mappings for the exponential similarity function. Random Feature Attention [PPY+21] uses random Fourier features to produce an approximate feature mapping but without provable approximation guarantees. Performer [CLD+20] provides a low dimensional approximate non-negative feature mapping $\phi'(\cdot)$ via positive orthogonal random features. It has provable approximation to the pairwise similarities, i.e., the maximum error $\max_{i,j\in[n]} \left|\langle \phi'(\boldsymbol{q}_i), \phi'(\boldsymbol{k}_j) \rangle - \exp(\langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle)\right|$ is small. However, the dimension of their feature mapping has to grow exponentially in $\|\boldsymbol{q}_i\|_2^2$ and $\|\boldsymbol{k}_j\|_2^2$ to have a small error. In other words, consider a single query $\boldsymbol{q}_i$ and two keys $\boldsymbol{k}_j$ and $\boldsymbol{k}_{j'}$ such that all $\|\boldsymbol{q}_i\|_2, \|\boldsymbol{k}_j\|_2, \|\boldsymbol{k}_{j'}\|_2 \le R$, then $\exp(\langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle)/\exp(\langle \boldsymbol{q}_i, \boldsymbol{k}_{j'} \rangle) \le \exp(2R^2)$. Thus, the maximum relative probability masses that can be assigned while guaranteeing the approximation factor is limited by the dimension of the feature mapping used. In fact, a recent work [AS23a] implies that it is actually impossible to get above approximation for pairwise exponential similarity under Strong Exponential Time Hypothesis (SETH [IPZ01]) when the query and key vectors have large entries. Furthermore, it was observed empirically [CLD+20, HDLL22] (also see Figure 9.2) that there is a clear model quality drop when using Performer in comparison with the exact softmax attention.

Given barriers above, a natural question arises: *Does there exist a similarity kernel function that achieves similar model quality as softmax attention while also admitting proper approximation by a low-dimensional feature mapping?*

9.1.1 Our Contributions

Polynomial similarity kernel function of high degree. To tackle the first part of the above question, we explore the power of the polynomial kernel function $\sigma(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle^p$ for large even degrees $p \geq 4$ empirically for language modeling tasks. In particular, we look at the standard GPT-2 [RWC+19] architecture (from the small size to the large size) and the strongest known Transformer recipe (a.k.a. Transformer++) which is a common baseline model studied in many previous works as well [HDLL22, GD23, YWS+23]. We compare the models with vanilla softmax attention to the models that simply replace the attention mechanism with degree-p polynomial attention. We consider context lengths ranging from 512 to 32k. As shown in Figure 9.2 and our other empirical studies (see Section 9.4), for autoregressive pre-training metrics (perplexity) and few-shot evaluations that we studied, the models with degree-p polynomial attention ($p \geq 4$) achieve comparable performance to the models with the vanilla softmax attention. In addition, we discuss the behavioral similarities between softmax attention and polynomial attention in Section 9.2.1 to provide more intuitions why they had similar empirical outcomes.

Approximate feature mapping for polynomial kernel. Unlike exponential kernel whose exact feature mapping has infinite dimension, the feature mapping of degree-p polynomial kernel over $\mathbb{R}^h$ has a finite feature mapping of dimension h^p (see e.g., [ANW14]). In practice, the head size h is usually 64, 128 or even 256 [CND+23]. Therefore, computing the exact feature mapping for $p \geq 4$ is still expensive. To address this issue, we use sketching to compute a low-dimensional approximate feature mapping ϕ' such that $\langle \phi'(\boldsymbol{x}), \phi'(\boldsymbol{y}) \rangle \approx \langle \boldsymbol{x}, \boldsymbol{y} \rangle^p$. Sketching polynomial kernels [ANW14, AKK+20, SWYZ21a, MSW19] has been extensively studied in the literature, and the techniques are used in many applications such as kernel regression [SWYZ21a], kernel PCA [ANW14], evaluating element-wise matrix functions [HAS20], etc. However, though $\langle \boldsymbol{x}, \boldsymbol{y} \rangle^p$ is guaranteed to be non-negative for even integer p, none of the approximate feature mappings provided by previous works guarantees $\langle \phi'(\boldsymbol{x}), \phi'(\boldsymbol{y}) \rangle \geq 0$. This is undesired in practice since the original normalized attention weights

$$\frac{\langle \boldsymbol{q}_i, \boldsymbol{k}_1 \rangle^p}{\sum_{j \in [n]} \langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle^p}, \frac{\langle \boldsymbol{q}_i, \boldsymbol{k}_2 \rangle^p}{\sum_{j \in [n]} \langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle^p}, \cdots, \frac{\langle \boldsymbol{q}_i, \boldsymbol{k}_n \rangle^p}{\sum_{j \in [n]} \langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle^p}$$

naturally represent a probability distribution, but the property does not hold when there exists some negative attention weight $\langle \phi'(\boldsymbol{q}_i), \phi'(\boldsymbol{k}_j) \rangle$. More importantly, previous work [CLD+20, KVPF20] found that negative attention weights make the training process unstable, potentially causing non-convergence. To address this issue, we extend the construction of [AKK+20] and develop an approximate feature mapping with desired non-negativity property.

Theorem 9.1.1. *Let $p \geq 2$ be an even integer, $\varepsilon \in (0, 0.5)$ be an error parameter. Let h be the dimension of the vectors to be mapped. There is a randomized feature mapping $\phi' : \mathbb{R}^h \to \mathbb{R}^{r^2}$ for $r = \Theta(p\varepsilon^{-2} \log 1/\delta)$, such that given any set of vectors $\{\boldsymbol{q}_i \in \mathbb{R}^h\}_{i \in [n]}, \{\boldsymbol{k}_j \in \mathbb{R}^h\}_{i \in [n]}$:*

1. *$\forall i, j \in [n], \langle \phi'(\boldsymbol{q}_i), \phi'(\boldsymbol{k}_j)\rangle \geq 0$.*
2. *$\sum\limits_{i,j} |\langle \phi'(\boldsymbol{q}_i), \phi'(\boldsymbol{k}_j)\rangle - \langle \boldsymbol{q}_i, \boldsymbol{k}_j\rangle^p|^2 \leq \varepsilon^2 \sum\limits_{i,j} \|\boldsymbol{q}_i\|_2^{2p} \|\boldsymbol{k}_j\|_2^{2p}$ holds with probability $1-\delta$.*
3. *Computing $\phi'(\boldsymbol{x})$ only requires $p/2$ matrix-vector multiplications of matrix size $h \times r$, $(p/2-2)$ matrix-vector multiplications of matrix size $r \times r$, $(p/2-1)$ Hadamard products of r-dimensional vectors, and 1 self-Kronecker product of an r-dimensional vector.*

The first property above is the desired non-negativity property that we discussed earlier. We achieve this property by using a simple "self-tensoring" technique. Our result is stated in Theorem 9.2.4. The second property states our error bound. The third property implies that the computation of $\phi'(\cdot)$ only requires a small number of standard matrix/vector operations which can be implemented to run quickly on accelerators (GPUs/TPUs).

Inspired by the literature of learned sketches [HIKV19, AIV19], we also propose a heuristic which replaces each random projection matrix used in $\phi'(\cdot)$ constructed in Theorem 9.1.1 with a comparable size learnable multi-layer dense network. Since each random matrix used in $\phi'(\cdot)$ has size only $h \times r$ or $r \times r$, the number of parameters that we add to the model is negligible in comparison to the model size. We observe significant model quality improvements (see Figure 9.2) by learning the sketches through training instead of using randomly sampled sketches.

Block-based lower triangular multiplication for handling causal masks. Another bottleneck in applying attention linearization techniques in training transformer models with causal masking on long contexts is to handle a huge number of sequential gradient updates due to RNN-style sequential dependencies [HDLL22]. When causal masking is applied, the attention between the query $\boldsymbol{q}_i$ and the key $\boldsymbol{k}_j$ is masked out when $j > i$ (i.e., the j-th token appears later). More precisely, $\mathsf{Attn}(\boldsymbol{q}_i, \{\boldsymbol{k}_j\}, \{\boldsymbol{v}_j\}) =$

$$\sum_{j\in[i]} \frac{\sigma(\boldsymbol{q}_i, \boldsymbol{k}_j)}{\sum_{j'\in[i]} \sigma(\boldsymbol{q}_i, \boldsymbol{k}_{j'})} \cdot \boldsymbol{v}_j^{\mathrm{T}} = \frac{\phi(\boldsymbol{q}_i)^{\mathrm{T}} \cdot \sum_{j\in[i]} \phi(\boldsymbol{k}_j) \cdot \boldsymbol{v}_j^{\mathrm{T}}}{\phi(\boldsymbol{q}_i)^{\mathrm{T}} \cdot \sum_{j'\in[i]} \phi(\boldsymbol{k}_{j'})}.$$

During the training, to compute the output of the attention mechanism in time linear in context length, one has to compute the prefix sums $\sum_{j\in[i]} \phi(\boldsymbol{k}_j) \cdot \boldsymbol{v}_j^{\mathrm{T}}$ for all i and then multiply the i-th prefix sum with the corresponding vector $\phi(\boldsymbol{q}_i)^{\mathrm{T}}$. This RNN-style sequential state updates make the training process fail in fully utilizing the parallelism strength of modern accelerators. To resolve above issue, we propose a general block-based approach to compute $\mathsf{lt}_{\triangle}(\boldsymbol{A} \cdot \boldsymbol{B}^{\mathrm{T}}) \cdot \boldsymbol{C}$[4] for arbitrary matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ without materializing $\boldsymbol{A} \cdot \boldsymbol{B}^{\mathrm{T}}$, and it only requires a small number of prefix updates. By working more carefully with our block-based approach, we observe that instead of using the approximate polynomial attention weight via approximate feature mapping, we are able to compute the exact polynomial attention weight between $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ if the i-th token and the j-th token are close

[4] $\mathsf{lt}_{\triangle}(\boldsymbol{M})$ denotes the matrix obtained by only keeping the lower triangular entries of $\boldsymbol{M}$ and zeroing the rest of the entries.

in position. After applying exact polynomial attention weight for local tokens, we see improvements in model qualities (see Figure 9.2, Section 9.4).

Empirical studies. We empirically evaluate all our approaches. The models equipped with high degree polynomial attention and sketched polynomial attention achieve comparable or better quality on all our evaluation metrics in comparison with models equipped with vanilla softmax attention, and achieve significantly better quality than models with approximate softmax attention provided by Performer [CLD+20]. For GPT-2 style small size models, the models with sketched polynomial attention achieve 2x speedup in comparison with FlashAttention [DFE+22, Dao23] of the fastest configuration for 32k context length. Notice that we achieve such speed-up without applying any advanced I/O aware optimization techniques. We believe that our running time can be further reduced by optimizing the implementation in a more careful manner.

9.1.2 Other Notation

Given a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times m}$, we use $\mathbf{m}_i \in \mathbb{R}^m$ to denote the i-th row of $\boldsymbol{M}$. We also abuse the notation to use $\boldsymbol{M}$ to indicate the set of vectors $\{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n\}$. We use $\boldsymbol{M}_{i,j}$ to denote the entry at the i-th row and j-th column of $\boldsymbol{M}$. We use $\boldsymbol{M}^p$ to indicate raising each entry of $\boldsymbol{M}$ to the power p. Let $f : \mathbb{R}^m \to \mathbb{R}^k$ be an arbitrary function over vectors, we use $f(\boldsymbol{M}) \in \mathbb{R}^{n\times k}$ to denote the matrix obtained by replacing the i-th row of $\boldsymbol{M}$ with $f(\mathbf{m}_i)$. Given vectors $\boldsymbol{a} = (a_1, a_2, \cdots, a_m)$ and $\boldsymbol{b} = (b_1, b_2, \cdots, b_m)$, $\boldsymbol{a} * \boldsymbol{b} = (a_1b_1, a_2b_2, \cdots, a_mb_m)$ denotes the entrywise product (Hadamard product), $\boldsymbol{a} \otimes \boldsymbol{b} = (a_1b_1, a_1b_2, \cdots, a_1b_m, a_2b_1, a_2b_2, \cdots, a_2b_m, \cdots, a_mb_m)$ denotes the Kronecker product, and $\operatorname{diag}(\boldsymbol{a})$ denotes a diagonal matrix where the i-th diagonal entry is a_i. $\boldsymbol{A}*\boldsymbol{B}$ denotes the entrywise product between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. We define "self-tensoring" $\boldsymbol{a}^{\otimes p} \coloneqq \boldsymbol{a} \otimes \boldsymbol{a}^{\otimes(p-1)} \in \mathbb{R}^{m^p}$ where $\boldsymbol{a}^{\otimes 1} \coloneqq \boldsymbol{a}$. $\boldsymbol{A}^{\otimes p}$ indicates replacing each row $\boldsymbol{a}_i$ of $\boldsymbol{A}$ with $\boldsymbol{a}_i^{\otimes p}$. $\mathsf{lt}_{\triangle}(\boldsymbol{M})$ denotes the matrix obtained by only keeping the lower-triangular entries of $\boldsymbol{M}$ and zeroing the remaining. $\mathbf{1}_m \in \mathbb{R}^m$ denotes an all-one vector.

9.2 Polynomial Attention and Approximation

We discuss the polynomial attention in more detail in Section 9.2.1 and introduce the sketching techniques (Section 9.2.2, 9.2.3) for efficiently approximating the polynomial attention. We ignore causal masking in this section, and present how to efficiently handle causal masking in Section 9.3.

9.2.1 Softmax versus Normalized Polynomial

Let us revisit the softmax attention. Given sets of query, key vectors $\boldsymbol{Q} = \{\boldsymbol{q}_i\}_{i\in[n]}, \boldsymbol{K} = \{\boldsymbol{k}_i\}_{i\in[n]} \subset \mathbb{R}^h$, and scaling parameter $\beta > 0$, bias parameter $\alpha \in \mathbb{R}$, the normalized softmax attention weight

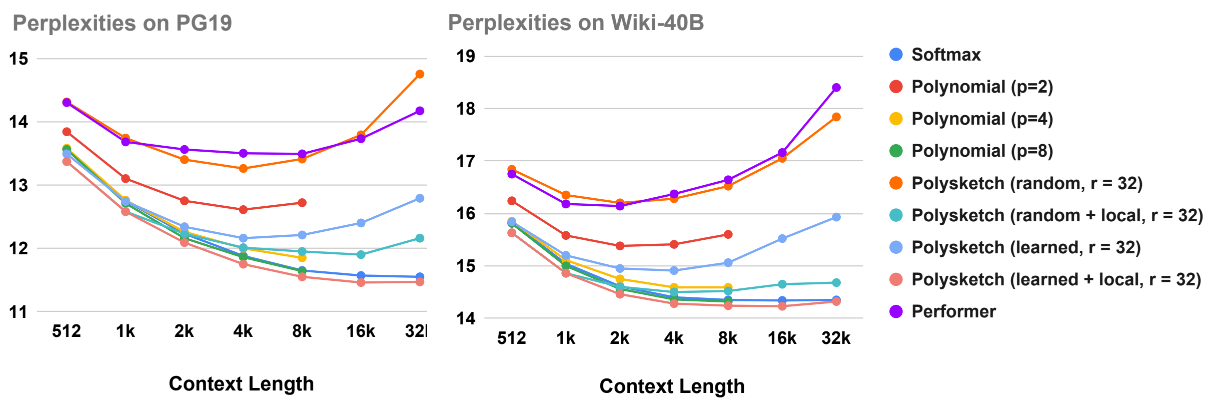


Figure 9.2: **Pre-training metric (perplexities). Lower is better.** GPT-2 small style models with various attention mechanisms are trained on PG-19 and Wiki-40B datasets at different context lengths up to 32k. Each batch contains 1M tokens in total. Polynomial attention with $p \geq 4$ has comparable model quality as softmax attention but OOM'ed when context length >8k. Polysketch attention (equipped with learned sketches (Section 9.2.3) + local exact polynomial attention (Section 9.3.2)) consistently outperforms all other mechanisms. The parameter r denotes the sketch size (see formal definition in Section 9.2.2).

between $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ is:

$$\boldsymbol{A}_{i,j} = \frac{\exp\left(\langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle / \beta - \alpha\right)}{\sum_{j' \in [n]} \exp\left(\langle \boldsymbol{q}_i, \boldsymbol{k}_{j'} \rangle / \beta - \alpha\right)}.$$

Note $\boldsymbol{A}_{i,j}$ is invariant in α. In practice, α is usually chosen to be $\max_{j' \in [n]} \langle \boldsymbol{q}_i, \boldsymbol{k}_{j'} \rangle / \beta$ to make the computation of both numerator and denominator numerically stable. β is a smoothness factor. When $\beta \to \infty$, then $\boldsymbol{A}_{i,j} \to 1/n$, i.e., the i-th row of $\boldsymbol{A}$ indicates a uniform distribution over all $j \in [n]$. When $\beta \to 0$, then $\boldsymbol{A}_{i,j} \to \begin{cases} 0 & \langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle \neq \max_{j' \in [n]} \langle \boldsymbol{q}_i, \boldsymbol{k}_{j'} \rangle \\ 1 \,/\, a & \langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle = \max_{j' \in [n]} \langle \boldsymbol{q}_i, \boldsymbol{k}_{j'} \rangle \end{cases}$ where a is the number of j satisfying $\langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle = \max_{j' \in [n]} \langle \boldsymbol{q}_i, \boldsymbol{k}_{j'} \rangle$, i.e., the i-th row of $\boldsymbol{A}$ indicates a uniform distribution only over j that provides the maximum inner product. In general, β is chosen to be $\sqrt{h}$ [VSP+17].

Interestingly, normalized polynomial function has a similar behavior of the interpolation nature between the uniform distribution and the argmax distribution discussed above. In particular, let p be an even integer and consider the following normalized weight computed between $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$:

$$\frac{\left(\left(\langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle + \alpha\right)/\beta\right)^p}{\sum_{j' \in [n]} \left(\left(\langle \boldsymbol{q}_i, \boldsymbol{k}_{j'} \rangle + \alpha\right)/\beta\right)^p}. \tag{9.1}$$

It is clear the weight is invariant for different β. Choosing a proper β can make both numerator and denominator fall in a reasonable range and make the computation numerically stable. When

$\alpha \to \infty$, the weight is close to $1/n$, i.e., these weights provide a uniform distribution. When $\alpha \geq -\min_{j' \in [n]} \langle \boldsymbol{q}_i, \boldsymbol{k}_{j'} \rangle$ and $p \to \infty$, the weight is close to 0 if $\langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle$ does not provide the maximum inner product, and the weight is close to $1/a$ otherwise, where a is the number of $\boldsymbol{k}_j$ that provides the maximum inner product.

Observe that if $\langle \boldsymbol{q}_i, \boldsymbol{1}_h \rangle = \langle \boldsymbol{k}_j, \boldsymbol{1}_h \rangle = 0$ for all $i, j \in [n]$, i.e., entries of $\boldsymbol{q}_i, \boldsymbol{k}_j$ always have mean 0, then we have $\forall i, j \in [n]$, $(\langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle + \alpha)/\beta = \left\langle \boldsymbol{q}'_i, \boldsymbol{k}'_j \right\rangle$, where $\boldsymbol{q}'_i = \boldsymbol{q}_i/\sqrt{\beta} + \sqrt{\alpha/(\beta h)} \cdot \boldsymbol{1}_h$, and $\boldsymbol{k}'_j = \boldsymbol{k}_j/\sqrt{\beta} + \sqrt{\alpha/(\beta h)} \cdot \boldsymbol{1}_h$, i.e., $\boldsymbol{q}'_i$ and $\boldsymbol{k}'_j$ are obtained by applying the same rescaling and bias to $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ respectively. Motivated by the above observation, we slightly tweak Equation 9.1 by applying an additional layer normalization[5] [BKH16] to $\{\boldsymbol{q}_i\}, \{\boldsymbol{k}_j\}$, this gives the normalized degree-p polynomial attention weight matrix $\boldsymbol{A}^{(p)}$ considered here:

$$\boldsymbol{A}^{(p)}_{i,j} = \frac{\langle \boldsymbol{q}'_i, \boldsymbol{k}'_j \rangle^p}{1 + \sum_{j' \in [n]} \langle \boldsymbol{q}'_i, \boldsymbol{k}'_{j'} \rangle^p}$$

where $\boldsymbol{q}'_i, \boldsymbol{k}'_j$ are obtained by applying the layer normalization layer to $\boldsymbol{q}_i, \boldsymbol{k}_j$ respectively. Unlike softmax attention matrix, it is possible that the term $\sum_{j' \in [n]} \langle \boldsymbol{q}'_i, \boldsymbol{k}'_{j'} \rangle^p$ is (close to) 0. We add 1 to the denominator to avoid dividing by zero. Given value vectors $\boldsymbol{V} = \{\boldsymbol{v}_i\}_{i \in [n]} \subset \mathbb{R}^h$, the full degree-$p$ polynomial attention $\mathsf{Attn}^{(p)}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{A}^{(p)} \cdot \boldsymbol{V} = \boldsymbol{D}^{-1} \cdot (\boldsymbol{Q}' \boldsymbol{K}'^{\top})^p \cdot \boldsymbol{V}$, where $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{1}_n + (\boldsymbol{Q}' \boldsymbol{K}'^{\top})^p \boldsymbol{1}_n)$. In the rest of the chapter, we abuse notation between $\boldsymbol{Q}, \boldsymbol{K}$ and $\boldsymbol{Q}', \boldsymbol{K}'$, and only consider $\boldsymbol{Q}, \boldsymbol{K}$ after layer normalization.

As presented in Figure 9.2 and other experiments in Section 9.4, the models with the degree-p polynomial attention described above achieve comparable model quality as vanilla softmax attention on all metrics as long as $p \geq 4$.

9.2.2 Random Sketches for Polynomial Attention with Theoretical Guarantees

To compute $\mathsf{Attn}^{(p)}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$, we only need to compute $(\boldsymbol{Q}\boldsymbol{K}^{\top})^p \cdot \boldsymbol{V}$ and $(\boldsymbol{Q}\boldsymbol{K}^{\top})^p \cdot \boldsymbol{1}_n$. Let us only focus on computing $(\boldsymbol{Q}\boldsymbol{K}^{\top})^p \cdot \boldsymbol{V}$ since we can handle $(\boldsymbol{Q}\boldsymbol{K}^{\top})^p \cdot \boldsymbol{1}_n$ in the same way. Due to a well-known fact $\forall \boldsymbol{x}, \boldsymbol{y}$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle^p = \langle \boldsymbol{x}^{\otimes p}, \boldsymbol{y}^{\otimes p} \rangle$, we have $(\boldsymbol{Q}\boldsymbol{K}^{\top})^p \boldsymbol{V} = \boldsymbol{Q}^{\otimes p} (\boldsymbol{K}^{\otimes p})^{\top} \cdot \boldsymbol{V}$. If we reorder the computation and compute $(\boldsymbol{K}^{\otimes p})^{\top} \cdot \boldsymbol{V}$ first, we are able to compute $\boldsymbol{Q}^{\otimes p} \cdot (\boldsymbol{K}^{\otimes p})^{\top} \cdot \boldsymbol{V}$ in $O(nh^{p+1})$ time which is linear in the context length n. However, the h^{p+1} dependence is still expensive as we explained in Section 9.1.1. Thus, we resort to approximating $\boldsymbol{Q}^{\otimes p} (\boldsymbol{K}^{\otimes p})^{\top}$ using sketching techniques, which we formally describe ahead. We first state the definition of a sketch that has the "Approximate Matrix Multiplication (AMM)" guarantee.

Definition 9.2.1 (Approximate Matrix Multiplication [Woo14])**.** Given parameters n, h and p, a randomized sketching matrix $\boldsymbol{S} \in \mathbb{R}^{h^p \times r}$ has the (ε, p)-AMM property if given any two $n \times h$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, with probability $\geq 9/10$ over the randomized sketching matrix $\boldsymbol{S}$, we have that

[5]Layer normalization shifts the entries of the input vector to make them have mean 0 and learns a suitable bias during training.

$\|(A^{\otimes p}S)(B^{\otimes p}S)^{\mathrm{T}} - A^{\otimes p}(B^{\otimes p})^{\mathrm{T}}\|_{\mathrm{F}} \leq \varepsilon \|A^{\otimes p}\|_{\mathrm{F}}\|B^{\otimes p}\|_{\mathrm{F}}$.

The parameter r above is referred to as the *sketch size*. Two important properties of a sketching distribution are (i) the sketch size r as a function of the accuracy parameter ε and (ii) the time required to compute $A^{\otimes p}S$ given an arbitrary matrix A. Ideally, we want the matrix S to have a structure such that $A^{\otimes p}S$ can be computed without realizing the large matrix $A^{\otimes p}$. [AKK+20] gave constructions of different sketches that have both the properties that the sketch size r is small and the matrix $A^{\otimes p}S$ can be computed quickly. We describe the main properties of one of their sketches below and explain how to compute $A^{\otimes p}S$.

Theorem 9.2.2 ([AKK+20]). *Given p and ε, there is a sketching matrix S with $r = \Theta(p/\varepsilon^2)$ columns such that S satisfies the (ε, p)-AMM property (Definition 9.2.1). Given an arbitrary vector $a \in \mathbb{R}^h$, computing $(a^{\otimes p})^\top S$ only requires p matrix-vector multiplications of matrix size $h \times r$, $(p-2)$ matrix-vector multiplications of matrix size $r \times r$, and $(p-1)$ Hadamard products of r-dimensional vectors.*

To compute $A^{\otimes p}S$, we only need to compute $(a_i^{\otimes p})^\top S$ for each row a_i of A. The number of matrix-vector multiplications and Hadamard products scales linearly in n. Let us focus on the construction of the sketch described in Theorem 9.2.2. We now explain how the sketch computation works for $p = 2$ and how it is extended to general values of p that are powers of 2. Let $G_1 \in \mathbb{R}^{h\times r}$ and $G_2 \in \mathbb{R}^{h\times r}$ denote two independently sampled random Gaussian matrices, i.e., each entry is drawn independently from a standard Gaussian distribution. Then the outcome of applying the sketch on $A^{\otimes 2}$ is $A^{\otimes 2}S = \sqrt{1/r} \cdot [(AG_1) * (AG_2)]$. The construction extends to all p that are powers of 2 in a recursive way. PolySketchWithNegativity(A, r, p) (Algorithm 9.1) shows how to compute $A^{\otimes p}S$ in general.

Algorithm 9.1: PolySketchWithNegativity

```
   Input: $A \in \mathbb{R}^{k\times m}, r, p$
   // Implementation of Theorem 9.2.2 [AKK+20].
   // The output is $A^{\otimes p}S$.
1  if $p = 1$ then
2  |   return $A$
3  end
4  $M_1 \leftarrow$ PolySketchWithNegativity$(A, r, p/2)$
5  $M_2 \leftarrow$ PolySketchWithNegativity$(A, r, p/2)$
6  Sample Gaussian matrices $G_1, G_2$, each of $r$ columns
7  return $\sqrt{1/r} \cdot [(M_1G_1) * (M_2G_2)] \in \mathbb{R}^{k\times r}$
```

The polynomial sketch can be used to approximate the matrix $(QK^{\mathrm{T}})^p = Q^{\otimes p}(K^{\otimes p})^{\mathrm{T}}$ with $(Q^{\otimes p}S)(K^{\otimes p}S)^{\mathrm{T}}$. However, one issue is that they do not preserve nonnegativity: while for even p, the entries of the matrix $(QK^{\mathrm{T}})^p$ are nonnegative, the entries of the matrix $(Q^{\otimes p}S)(K^{\otimes p}S)^{\mathrm{T}}$ can be negative. This is not desired as discussed in Section 9.1.1. In the following, we propose a simple approach to address this negativity issue.

Algorithm 9.2: PolySketchNonNegative

```
  Input: $A \in \mathbb{R}^{k\times m}, r, p$
  // Our approach based on Theorem 9.2.4.
  // The output computes $\phi'(A) = (A^{\otimes(p/2)}S)^{\otimes 2}$ where $\phi'(\cdot)$ is the same mapping
     as mentioned in Theorem 9.1.1.
1 $M \leftarrow$ PolySketchWithNegativity$(A, r, p/2)$ return $M^{\otimes 2} \in \mathbb{R}^{k\times r^2}$.
```

Consider two arbitrary vectors $\boldsymbol{a}, \boldsymbol{b}$, we can see that the dot product $\langle \boldsymbol{a}^{\otimes 2}, \boldsymbol{b}^{\otimes 2}\rangle = \langle \boldsymbol{a}, \boldsymbol{b}\rangle^2 \geq 0$. Thus, given matrices $\boldsymbol{Q}^{\otimes(p/2)}\boldsymbol{S}$ and $\boldsymbol{K}^{\otimes(p/2)}\boldsymbol{S}$, consider the matrix $(\boldsymbol{Q}^{\otimes(p/2)}\boldsymbol{S})^{\otimes 2}((\boldsymbol{K}^{\otimes(p/2)}\boldsymbol{S})^{\otimes 2})^{\mathrm{T}}$. Since all the entries of the matrix are of the form $\langle \boldsymbol{a}^{\otimes 2}, \boldsymbol{b}^{\otimes 2}\rangle$ for some vectors $\boldsymbol{a}, \boldsymbol{b}$, all the entries of the matrix $(\boldsymbol{Q}^{\otimes(p/2)}\boldsymbol{S})^{\otimes 2}((\boldsymbol{K}^{\otimes(p/2)}\boldsymbol{S})^{\otimes 2})^{\mathrm{T}}$ are nonnegative as well. The "self-tensoring" trick ensures that all the entries in the approximate attention matrix are nonnegative at the cost of *squaring* the sketch size r.

Although $(\boldsymbol{Q}^{\otimes(p/2)}\boldsymbol{S})^{\otimes 2}((\boldsymbol{K}^{\otimes(p/2)}\boldsymbol{S})^{\otimes 2})^{\mathrm{T}}$ guarantees non-negative property, it is not clear whether it is still a good approximation to $(\boldsymbol{Q}\boldsymbol{K}^{\top})^p$ given that $\boldsymbol{S}$ is a polynomial sketch for degree $p/2$. One of our technical contributions is to prove that it still provides a good approximation when the sketching matrix $\boldsymbol{S}$ is constructed as in [AKK+20]. The key is Theorem 9.2.4 which shows that a degree $p/2$ polynomial sketch followed by "self-tensoring" gives a degree p polynomial sketch.

To state Theorem 9.2.4 properly, we need to briefly introduce the following concepts. The (ε, δ, t)-JL moment property is defined as follows. Given a scalar random variable X and $t \geq 1$, $\|X\|_{L^t}$ is defined to be $\mathbf{E}[|X|^t]^{1/t}$. $\|\cdot\|_{L^t}$ defines a norm over random variables defined over the same sample space and in particular satisfies $\|X+Y\|_{L^t} \leq \|X\|_{L^t} + \|Y\|_{L^t}$.

Definition 9.2.3 (JL-moment property [Woo14])**.** Given $\varepsilon, \delta \geq 0, t \geq 1$, a random matrix $\boldsymbol{S}^{m\times r}$ has the (ε, δ, t)-JL moment property if for any $\boldsymbol{x} \in \mathbb{R}^m$ with $\|\boldsymbol{x}\|_2 = 1$, $\left\|\|\boldsymbol{x}^{\mathrm{T}}\boldsymbol{S}\|_2^2 - 1\right\|_{L^t} \leq \varepsilon \cdot \delta^{1/t}$.

Theorem 9.2.4. *Let $\boldsymbol{S} \in \mathbb{R}^{h^{p/2}\times r}$ be a random sketch satisfying the (ε, δ, t)-JL moment and $(\varepsilon, \delta, 2t)$-JL moment properties for some even integer t. Given matrices $\boldsymbol{C}, \boldsymbol{D}$ with $h^{p/2}$ columns, $\|(\boldsymbol{C}\boldsymbol{S})^{\otimes 2}((\boldsymbol{D}\boldsymbol{S})^{\otimes 2})^{\mathrm{T}} - \boldsymbol{C}^{\otimes 2}(\boldsymbol{D}^{\otimes 2})^{\mathrm{T}}\|_{\mathrm{F}} \leq \sqrt{5}\varepsilon\|\boldsymbol{C}^{\otimes 2}\|_{\mathrm{F}}\|\boldsymbol{D}^{\otimes 2}\|_{\mathrm{F}}$ holds with probability $\geq 1-\delta$,*

We first note the following fact: If $\boldsymbol{S}$ has (ε, δ, t)-JL moment property, then for any two arbitrary vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we have that $\|\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{x}, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{y}\rangle - \langle \boldsymbol{x}, \boldsymbol{y}\rangle\|_{L^t} \leq \varepsilon\delta^{1/t}\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2$. For a proof see Lemma 9 from [AKK+20].

Proof of Theorem 9.2.4. Let $\boldsymbol{c}_i$ denote the i-th row of $\boldsymbol{C}$ and $\boldsymbol{d}_j$ denote the j-th row of $\boldsymbol{D}$. Then the (i,j)-th entry of the matrix $\boldsymbol{C}^{\otimes 2}(\boldsymbol{D}^{\otimes 2})^{\mathrm{T}}$ is equal to $\langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2$. Similarly, the (i,j)-th coordinate of the matrix $(\boldsymbol{C}\boldsymbol{S})^{\otimes 2}((\boldsymbol{D}\boldsymbol{S})^{\otimes 2})^{\mathrm{T}}$ is equal to $\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle^2$ and therefore

$$\|(\boldsymbol{C}\boldsymbol{S})^{\otimes 2}((\boldsymbol{D}\boldsymbol{S})^{\otimes 2})^{\mathrm{T}} - \boldsymbol{C}^{\otimes 2}(\boldsymbol{D}^{\otimes 2})^{\mathrm{T}}\|_{\mathrm{F}}^2 = \sum_{i,j}(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle^2 - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2)^2.$$

Recall that given an integer $t \geq 1$, for a random variable $\boldsymbol{X}$, we define $\|\boldsymbol{X}\|_{L^t}$ as $\mathbf{E}[|\boldsymbol{X}|^t]^{1/t}$. Also note that $\|\boldsymbol{X}\|_{L^t}$ is a norm over the random variables and in-particular satisfies the triangle inequality. Now,

$$\begin{aligned}\|\|(\boldsymbol{C}\boldsymbol{S})^{\otimes 2}((\boldsymbol{D}\boldsymbol{S})^{\otimes 2})^{\mathrm{T}} - \boldsymbol{C}^{\otimes 2}(\boldsymbol{D}^{\otimes 2})^{\mathrm{T}}\|_{\mathrm{F}}\|_{L^t} &= \|\|(\boldsymbol{C}\boldsymbol{S})^{\otimes 2}((\boldsymbol{D}\boldsymbol{S})^{\otimes 2})^{\mathrm{T}} - \boldsymbol{C}^{\otimes 2}(\boldsymbol{D}^{\otimes 2})^{\mathrm{T}}\|_{\mathrm{F}}^2\|_{L^{t/2}}^{1/2}\\ &= \|\sum_{i,j}(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle^2 - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2)^2\|_{L^{t/2}}^{1/2}\\ &\leq (\sum_{i,j}\|(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle^2 - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2)^2\|_{L^{t/2}})^{1/2}\end{aligned}$$

where we used the triangle inequality of $\|\cdot\|_{L^t}$ in the last inequality. Now consider a single term $\|(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle^2 - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2)^2\|_{L^{t/2}}$. First, we have

$$\begin{aligned}&(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle^2 - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2)^2\\ &= (\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle + \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^2(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^2\\ &= (\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle + 2\langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^2(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^2\\ &\leq (1+C)(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^4 + 4(1+1/C)\langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^2\end{aligned}$$

with probability 1 for any $C \geq 1$. Since both LHS and RHS are *non-negative* random variables, we obtain that

$$\begin{aligned}&\|(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle^2 - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2)^2\|_{L^{t/2}}\\ &\leq (1+C)\|(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^4\|_{L^{t/2}} + 4(1+1/C)\langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2\|(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^2\|_{L^{t/2}}.\end{aligned}$$

Now,

$$\begin{aligned}\|(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^4\|_{L^{t/2}} &= \|\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle\|_{L^{2t}}^4\\ &\leq \varepsilon^4\delta^{2/t}\|\boldsymbol{c}_i\|_2^4\|\boldsymbol{d}_j\|_2^4\end{aligned}$$

assuming that S has $(\varepsilon, \delta, 2t)$-JL moment property. We also have

$$\begin{aligned}\|(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle)^2\|_{L^{t/2}} &= \|\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle\|_{L^t}^2\\ &\leq \varepsilon^2\delta^{2/t}\|\boldsymbol{c}_i\|_2^2\|\boldsymbol{d}_j\|_2^2\end{aligned}$$

assuming that $\boldsymbol{S}$ has (ε, δ, t)-JL moment property. Overall, we get

$$\begin{aligned}&\|(\langle \boldsymbol{S}^{\mathrm{T}}\boldsymbol{c}_i, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{d}_j\rangle^2 - \langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2)^2\|_{L^{t/2}}\\ &\leq (1+C)\varepsilon^4\delta^{2/t}\|\boldsymbol{c}_i\|_2^4\|\boldsymbol{d}_j\|_2^4 + 4(1+1/C)\langle \boldsymbol{c}_i, \boldsymbol{d}_j\rangle^2\varepsilon^2\delta^{2/t}\|\boldsymbol{c}_i\|_2^2\|\boldsymbol{d}_j\|_2^2.\end{aligned}$$

Picking $C = 1/\varepsilon$ and assuming $\varepsilon \le 1/5$, we get that

$$\|(\langle S^{\mathrm{T}} c_i \rangle^2 - \langle c_i, d_j \rangle^2)^2\|_{L^{t/2}} \le 5\varepsilon^2 \delta^{2/t} \|c_i\|_2^4 \|d_j\|_2^4.$$

Thus, we have

$$\begin{aligned}\|\|(CS)^{\otimes 2}((DS)^{\otimes 2})^{\mathrm{T}} - C^{\otimes 2}(D^{\otimes 2})^{\mathrm{T}}\|_{\mathrm{F}}\|_{L^t} &\le \sqrt{5}\varepsilon\delta^{1/t}\sqrt{\sum_{i,j}\|c_i\|_2^4\|d_j\|_2^4} \\ &\le \sqrt{5}\varepsilon\delta^{1/t}\|C^{\otimes 2}\|_{\mathrm{F}}\|D^{\otimes 2}\|_{\mathrm{F}}.\end{aligned}$$

By using Markov's inequality, we obtain that with probability $\ge 1 - \delta$,

$$\|(CS)^{\otimes 2}((DS)^{\otimes 2})^{\mathrm{T}} - C^{\otimes 2}(D^{\otimes 2})^{\mathrm{T}}\|_{\mathrm{F}} \le \sqrt{5}\varepsilon\|C^{\otimes 2}\|_{\mathrm{F}}\|D^{\otimes 2}\|_{\mathrm{F}}. \qquad \square$$

Proof of Theorem 9.1.1. Results from Section 4 of [AKK+20] implies that the polynomial sketch S as mentioned in Theorem 9.2.2 for degree $p/2$ with sketch size $r = \Theta(p/\varepsilon^2)$ satisfies the requirements of Theorem 9.2.4. By plugging $Q^{\otimes(p/2)}$, $K^{\otimes(p/2)}$ into C, D of Theorem 9.2.4 respectively and scaling ε properly, we obtain $\|(Q^{\otimes(p/2)}S)^{\otimes 2}((K^{\otimes(p/2)}S)^{\otimes 2})^{\mathrm{T}} - (QK^\top)^p\|_{\mathrm{F}} \le \varepsilon\|Q^{\otimes p}\|_{\mathrm{F}}\|K^{\otimes p}\|_{\mathrm{F}}$ which concludes Theorem 9.1.1, i.e., the approximate feature mapping $\phi'(x) = ((x^{\otimes(p/2)})^\top S)^{\otimes 2} \in \mathbb{R}^{r^2}$ and $\phi'(Q), \phi'(K)$ can be efficiently computed using $\textsc{PolySketchNonNegative}(\cdot, r, p)$ (see Algorithm 9.1).

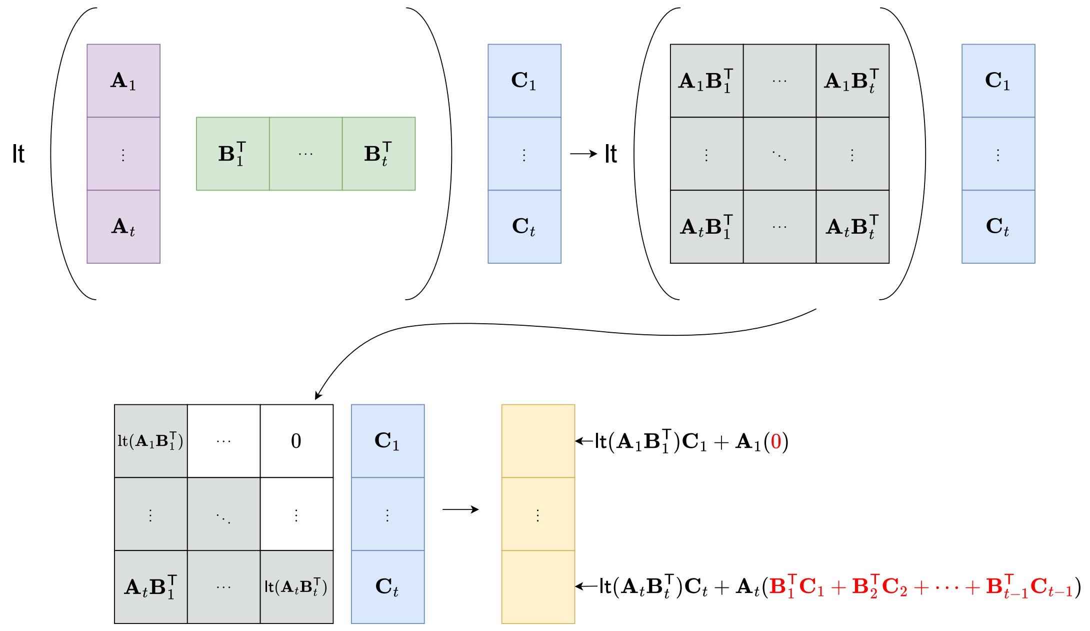
Using $\phi'(\cdot)$, we get the following approximate polynomial attention

$$\widetilde{\mathsf{Attn}}^{(p)}(Q, K, V) = \widetilde{D}^{-1}\phi'(Q)\phi'(K)^\top V,$$

where $\widetilde{D} = \mathrm{diag}(\mathbf{1}_n + \phi'(Q)\phi'(K)^\top \mathbf{1}_n)$. We call this attention mechanism Polysketch attention.

9.2.3 Learnable Sketches for Polynomial Attention

There are only $(p-2)$ random projections where each is introduced by a matrix multiplication with a small Gaussian matrix (G_1, G_2 in each recursion call in Algorithm 9.1) of size either $h \times r$ or $r \times r$ during the recursive computation of $\phi'(X) = \textsc{PolySketchNonNegative}(X, r, p)$ (Algorithm 9.2) for $X \in \mathbb{R}^{n\times h}$. Inspired by the literature of learned sketches [HIKV19, AIV19], a natural idea is to replace each random matrix G_1, G_2 in Algorithm 9.1 with learnable parameters. In practice, we found that replacing each of these random projections with a learnable *non-linear* transformation introduced by a dense neural network with size comparable to G_1, G_2 achieves a better model quality.



lt
$\mathbf{A}_1$
$\mathbf{A}_t$
$\mathbf{B}_1^\top$
$\mathbf{B}_t^\top$
$\mathbf{C}_1$
$\mathbf{C}_t$
lt
$\mathbf{A}_1\mathbf{B}_1^\top$
$\mathbf{A}_1\mathbf{B}_t^\top$
$\mathbf{A}_t\mathbf{B}_1^\top$
$\mathbf{A}_t\mathbf{B}_t^\top$
$\mathrm{lt}(\mathbf{A}_1\mathbf{B}_1^\top)$
0
$\mathrm{lt}(\mathbf{A}_t\mathbf{B}_t^\top)$
$\mathrm{lt}(\mathbf{A}_1\mathbf{B}_1^\top)\mathbf{C}_1 + \mathbf{A}_1(0)$
$\mathrm{lt}(\mathbf{A}_t\mathbf{B}_t^\top)\mathbf{C}_t + \mathbf{A}_t(\mathbf{B}_1^\top\mathbf{C}_1 + \mathbf{B}_2^\top\mathbf{C}_2 + \cdots + \mathbf{B}_{t-1}^\top\mathbf{C}_{t-1})$

Figure 9.3: **Block wise Lower Triangular Multiplication.** A_l, B_l, C_l are blocks of A, B, C. Each block has $b = n/t$ rows.

9.3 Handling Causal Masks

When considering causal masks, the Polysketch attention with respect to q_i is defined as

$$\sum_{j \le i} \frac{\langle \phi'(q_i), \phi'(k_j) \rangle}{1 + \sum_{j' \le i} \langle \phi'(q_i), \phi'(k_{j'}) \rangle} \cdot v_j^\top.$$

In this causal case, the full Polysketch attention can be written as

$$\widetilde{\mathsf{Attn}}^{(p)}(Q, K, V) = \widetilde{D}^{-1} \cdot \mathsf{lt}_{\triangle}(\phi'(Q)\phi'(K)^\top) \cdot V$$

where $\widetilde{D} = \mathrm{diag}(\mathbf{1}_n + \mathsf{lt}_{\triangle}(\phi'(Q)\phi'(K)^\top) \cdot \mathbf{1}_n)$. Therefore, computing $\mathsf{lt}_{\triangle}(\phi'(Q)\phi'(K)^\top) \cdot X$ for $X \in \{\mathbf{1}_n, V\}$ efficiently is crucial. In the next subsection, we present a block based algorithm to compute $\mathsf{lt}_{\triangle}(A \cdot B) \cdot C$ for arbitrary matrices $A, B \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{n \times k}$ in time linear in n, in which the number of sequentially dependent steps is small.

9.3.1 Fast Lower Triangular Multiplication

Let b be the block size and $t = n/b$ be the number of blocks where each block B_ℓ ($\ell \in [t]$) contains indices $\{(\ell - 1)b + 1, (\ell - 1)b + 2, \cdots, \ell \cdot b\}$. Let a_i, b_i, c_i denote the i-th row vector of A, B, C respectively. For each $\ell \in [t]$, let the rows of $A_\ell \in \mathbb{R}^{b \times m}$ consist of a_i where $i \in B_\ell$. We define sub-matrices $\boldsymbol{B}_\ell, \boldsymbol{C}_\ell$ of $\boldsymbol{B}, \boldsymbol{C}$ respectively in a similar way. For $\ell \in [t]$, let us compute $\boldsymbol{H}_\ell = \sum_{i \in B_\ell} \boldsymbol{b}_i \boldsymbol{c}_i^{\mathrm{T}}$. Let $\boldsymbol{Z}_l$ indicates the prefix sum: $\boldsymbol{Z}_\ell = \sum_{j<\ell} \boldsymbol{H}_j$. In addition, let us compute $\boldsymbol{P}_\ell = \mathsf{lt}_{\triangle}(\boldsymbol{A}_\ell \boldsymbol{B}_\ell^{\mathrm{T}})\boldsymbol{C}_\ell$ for each $\ell \in [t]$ in the direct way. For any $\ell \in [t]$, and any $i \in B_\ell$, if i is the i'-th index within the block B_ℓ, it is easy to verify that the i-th row of $\mathsf{lt}_{\triangle}(\boldsymbol{A}\boldsymbol{B}^{\mathrm{T}})\boldsymbol{C}$ can be obtained by $\boldsymbol{p} + \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{Z}_\ell$ where $\boldsymbol{p}$ is the i'-th row of $\boldsymbol{P}_\ell$. Figure 9.3 justifies the correctness of the above algorithm.

Since the prefix sum $\boldsymbol{Z}_t$ is over t matrices, the number of sequentially dependent steps is t. We can further reduce the number of sequential steps by using a parallel prefix sum algorithm [Ble90] to exploit the parallelism. In our implementation, we only use the sequential prefix sum algorithm. Computing all $\boldsymbol{P}_\ell$ requires $O(t \cdot b^2(m+k))$ time. Computing all $\boldsymbol{H}_\ell, \boldsymbol{Z}_\ell$ requires $O(t \cdot bmk)$ time. Computing all $\boldsymbol{A}_\ell \boldsymbol{Z}_\ell + \boldsymbol{P}_\ell$ requires $O(t \cdot bmk)$ time. Therefore, the overall running time is $O(nb(m+k))$. When we set b as a constant, the running time is linear in n. If we directly plug in $\phi'(\boldsymbol{Q}), \phi'(\boldsymbol{K}), \boldsymbol{V}$ (or $\mathbf{1}_n$) into $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ above respectively, we compute causal Polysketch attention in time linear in the context length, n.

Let us take another look using the above process to compute Polysketch attention, the matrix $\boldsymbol{P}_\ell$ actually corresponds to $\mathsf{lt}_{\triangle}(\phi'(\boldsymbol{Q})_\ell(\phi'(\boldsymbol{K})_\ell)^{\mathrm{T}})\boldsymbol{X}_\ell$ where $\boldsymbol{X}_\ell \in \{\boldsymbol{V}_\ell, \mathbf{1}_b\}$. $\phi'(\boldsymbol{Q})_\ell, \phi'(\boldsymbol{K})_\ell$ corresponds to approximate feature mapped query and key vectors within the block B_ℓ, and $\boldsymbol{V}_\ell$ corresponds to the value vectors in B_ℓ. Let $\boldsymbol{Q}_\ell, \boldsymbol{K}_\ell$ be the corresponding original query and key vectors in B_ℓ. One observation is that

$$\phi'(\boldsymbol{Q})_\ell(\phi'(\boldsymbol{K})_\ell)^{\mathrm{T}} = \boldsymbol{L}^{\otimes 2}(\boldsymbol{R}^{\otimes 2})^{\mathrm{T}} = \left(\boldsymbol{L}\boldsymbol{R}^{\mathrm{T}}\right)^2$$

where

$$\boldsymbol{L} = \textsc{PolySketchWithNegativity}(\boldsymbol{Q}_\ell, r, p/2) \in \mathbb{R}^{b \times r}$$
$$\boldsymbol{R} = \textsc{PolySketchWithNegativity}(\boldsymbol{K}_\ell, r, p/2) \in \mathbb{R}^{b \times r}.$$

Therefore, $\mathsf{lt}_{\triangle}(\phi'(\boldsymbol{Q})_\ell \phi'(\boldsymbol{K})_\ell^{\mathrm{T}})$ only takes $O(b^2 r)$ time instead of $O(b^2 r^2)$ time. The total time to compute Polysketch attention is $O(nb(r+h) + nr^2 h)$.

9.3.2 Applying Exact Attention Locally

We further observe that $\phi'(\boldsymbol{Q})_\ell \phi'(\boldsymbol{K})_\ell^{\mathrm{T}}$ is used to approximate $(\boldsymbol{Q}_\ell \boldsymbol{K}_\ell^{\mathrm{T}})^p$. We can actually compute $\boldsymbol{P}_\ell$ as $\mathsf{lt}_{\triangle}\left((\boldsymbol{Q}_\ell \boldsymbol{K}_\ell^{\mathrm{T}})^p\right)\boldsymbol{X}_\ell$. This means that when token i and j are within the same local block, we can use their exact polynomial attention weight instead of using the approximation. The time to compute $\mathsf{lt}_{\triangle}\left((\boldsymbol{Q}_\ell \boldsymbol{K}_\ell^{\mathrm{T}})^p\right)\boldsymbol{X}_\ell$ is at most $O(b^2 h)$. In this case, the total time to compute our Polysketch attention is at most $O(nh(b + r^2))$. When $b \le r^2$, the running time is $O(nhr^2)$. As observed by our empirical studies (see Figure 9.2 and Section 9.4, using exact polynomial attention weights inside each local block further improves the model quality.

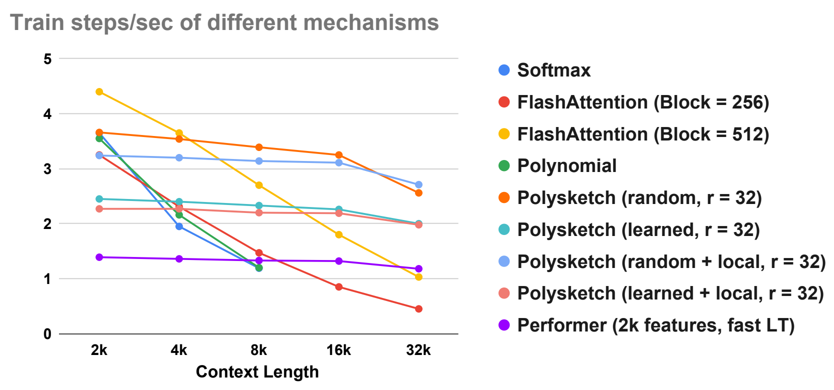


Figure 9.4: Training speed of models on PG-19 and Wiki-40B for different context lengths. Softmax and polynomial attentions OOM'ed when context length >8k.

9.4 Experiments

To evaluate the effectiveness of the polynomial attention and Polysketch attention mechanisms, we train language models of various sizes with different attention mechanisms and look at both pretraining metrics and the performances on downstream tasks. Our implementations of all models are written in JAX. In our experiments, we use a Pallas implementation [JAX23] of FlashAttention and a JAX implementation of Performer open-sourced by the authors [CLD+20]. All the experiments are conducted on 32 Google Cloud TPUs.

Models. For real world datasets, we train decoder-only models (only contain causal masked attention layers) of three different scales, mirroring the GPT-2 family [RWC+19]: Small, Medium and Large. For small scale models, we train models using context lengths from 512 to 32k. For medium scale models, we only train using context length 8k. For large scale models, we only train using context length 2k. The reason that we did not train longer context length for medium and large scale models is that non-kernel based attention mechanisms (softmax, polynomial) are too slow or go out of memory (OOM). If not specified otherwise, we use 10k warm up steps, 125k total training steps and a linear learning rate schedule. Depending on the original model scale, we also train kernel based attention models (Polysketch and Performer) with 0-3 additional layers, since these models are significantly faster than non-kernel based attention models, so we can afford to train larger models compared to vanilla softmax. It only slightly increases model sizes.

Attention Mechanisms. We train models with the following 4 categories of attention mechanisms: (i) Softmax, (ii) Polynomial ($p = 2, 4, 8$), (iii) Polysketch (approximating polynomial attention of $p = 4$) with variants enabling learned sketches (Section 9.2.3) or local exact polynomial attention (Section 9.3.2) or both, and (iv) Performer equipped with our lower triangular multiplication approach (Section 9.3.1) for handling causal masks. For both Performer and Polysketch, all attention

	C4 ↓	HellaSwag ↑		PIQA ↑		Physics ↑	
	Perplexity	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot
GPT-2 Small style, 100M-scale, 12 layers default, Context Length 8192, 125k training steps							
Softmax	17.81	30.2	27.8	64.6	63.2	27.5	27.5
Polynomial (degree 4)	18.18	28.6	28.4	64.2	**65.0**	27.5	31.0
Polynomial (degree 8)	17.77	29.8	29.8	62.2	64.0	23.1	26.2
Polysketch (learned, r = 64)	18.79	29.6	28.6	60.0	60.0	24.8	30.5
Polysketch (learned, 13 layers, r = 64)	18.47	28.4	29.4	62.0	62.6	27.5	31.8
Polysketch (learned + local, r = 64)	17.98	29.8	**30.6**	62.4	63.6	**30.1**	32.3
Polysketch (learned + local, 13 layers, r = 64)	**17.68**	29.0	29.0	62.6	64.2	20.5	27.0
Polysketch (learned, r = 32)	19.09	28.0	28.4	60.6	62.0	28.3	27.5
Polysketch (learned, 13 layers, r = 32)	19.50	28.4	29.0	61.6	64.6	27.9	33.1
Polysketch (learned + local, r = 32)	18.04	29.0	29.2	63.4	62.8	26.6	**35.8**
Polysketch (learned + local, 13 layers, r = 32)	17.72	**31.2**	30.4	**64.8**	64.6	27.9	31.8
GPT-2 Medium style, 300M-scale, 24 layers default, Context Length 8192, 125k training steps							
Softmax	**13.98**	35.8	**36.6**	67.0	67.2	30.5	25.7
Polynomial (degree 4)	14.29	35.8	36.0	65.8	67.6	27.5	28.8
Polynomial (degree 8)	14.14	37.0	**36.6**	65.4	65.6	**33.1**	27.5
Polysketch (learned, r = 64)	14.64	34.6	33.4	63.2	65.4	31.0	26.2
Polysketch (learned, 26 layers, r = 64)	14.49	34.8	34.4	65.2	66.6	28.4	24.9
Polysketch (learned + local, r = 64)	14.16	35.0	35.0	65.8	**68.6**	29.6	**34.5**
Polysketch (learned + local, 26 layers, r = 64)	**13.98**	35.8	35.4	66.4	**68.6**	27.0	33.6
Polysketch (learned, r = 32)	14.94	32.2	33.8	65.6	67.6	32.7	33.6
Polysketch (learned, 26 layers, r = 32)	14.73	32.8	35.2	65.0	65.2	28.3	31.8
Polysketch (learned + local, r = 32)	14.15	36.0	35.8	65.2	67.6	27.5	27.9
Polysketch (learned + local, 26 layers, r = 32)	14.00	**37.2**	35.4	**68.0**	67.6	23.1	29.6
GPT-2 Large style, 700M-scale, 36 layers default, Context Length 2048, 125k training steps							
Softmax	12.71	40.2	40.2	68.8	**71.4**	34.4	24.4
Polynomial (degree 4)	12.82	40.0	**40.6**	67.8	66.6	31.8	31.4
Polynomial (degree 8)	12.85	40.0	39.8	66.8	70.4	34.4	29.6
Polysketch (learned, 39 layers, r = 64)	12.83	**41.0**	39.4	68.6	68.8	33.6	36.6
Polysketch (learned + local, 39 layers, r = 64)	**12.70**	40.6	40.0	**69.0**	69.0	**38.4**	**37.1**
Polysketch (learned, 39 layers, r = 32)	12.98	39.4	40.4	68.6	67.6	33.6	27.0
Polysketch (learned + local, 39 layers, r = 32)	12.74	39.6	**40.6**	66.8	69.4	35.3	31.8

Table 9.1: We compare the accuracies(%, higher the better) of different models on three different Q/A tasks. HellaSwag and Physics tasks have 4 choices and PIQA task has 2 choices. We also report the perplexities (lower the better) on the validation split of C4 dataset. **Bolding** indicates the best model in the task, underlining indicates beating softmax attention.

heads share the same ϕ' within the same attention layer.

Hyperparameters. For FlashAttention, we try both block size 256 and 512[6]. For our fast lower triangular multiplication approach, we use $b = 1024$ for both Polysketch and Performer. We test both sketch sizes $r = 32$ and $r = 64$ for our Polysketch attention. We use 2048 features for Performer[7].

Pre-training metrics measurements (perplexities) over different context lengths. We train GPT-2 style small scale models equipped with different attention mechanisms on the Wiki-40B [GDVAR20] and PG-19 [RPJ+19] datasets with context length from 512 to 32k where each training batch contains 1M tokens. For all kernel based attentions (Performer and Polysketch), we use 13 layers instead of 12. The perplexity results are shown in Figure 9.2 and training latencies are shown in Figure 9.4. We observe that in the setting of 32k context length, Polysketch (learned + local, r=32) achieves **2x** speed-up in comparison with FlashAttention of the fastest setup.

Downstream tasks of language models. We train our models at different scales on the C4 dataset where each training batch contains 0.5M tokens. In Table 9.1, we report the perplexity on the validation split of C4 dataset and 0-shot and 5-shot accuracies on a random sample of 500 examples of HellaSwag [ZHB+19], 500 examples of PIQA [BZB+20] and on the full Physics question answering dataset [WW]. In addition to training models using 125k steps, we also train models with 30k steps to observe how the performance of attention mechanisms evolve with increasing number of total tokens trained on. As observed from Table 9.1, Polysketch attention has a comparable performance and sometimes outperforms softmax attention.

9.5 Conclusions and Future Work

In this work, we empirically studied the performance of using high degree polynomial attention instead of softmax attention in training decoder-only models for language modeling tasks. Our empirical study shows that the polynomial attention can achieve a similar model quality as the vanilla softmax attention when degree $p \geq 4$. Then we developed an efficient approximate polynomial attention via polynomial sketching techniques which can be computed in linear time of context length with provable approximation guarantees. In addition, we presented a fast block based lower triangular matrix multiplication algorithm which can significantly boost the training time of any kernel based attention in the decoder based models.

There are several potential directions for future works.

[6]We find a speed-up increasing the default 128 block size to 256 and 512 under our experimental setting. When increasing the block size to 1024, FlashAttention ran out of memory under our empirical setup.

[7]When using 4096 features, Performer ran out of memory in our experiments.

1. Although we only empirically studied the performance of decoder-only models with polynomial attention for language modeling tasks, it is interesting to explore the potentials of encoder models with polynomial attention, and to understand whether it can be used in other fields such as vision.
2. In this work, empirically we mainly focus on reducing the training latency. The benefits of linear transformers also transfer to inference as the KV cache sizes are independent of the context length. The exact inference improvements using linear transformers have to be explored more thoroughly.

Part II

The Streaming Setting

Chapter 10

Pseudorandom Hashing for Space-bounded Computation with Applications in Streaming

10.1 Introduction

Space-efficient algorithms are a central theme in computer science. In many cases, the best such algorithms are *randomized*, which raises the question of how to obtain the random bits. Nisan's classical pseudorandom generator [Nis92] shows that it is possible to expand a small random seed into a longer pseudorandom string that is essentially as good as full randomness if used in a space-bounded computation. In the context of *streaming algorithms* Nisan's generator has been used not only to reduce the need for random bits, but also because *storing* the seed allows us to re-create random values when they are needed, which is essentially a type of hashing [Cha02, Ind06, AGM12, DKS10, JST11, GKMS03, KNW10, FIS05]. Ideally we would like the space for the seed to be smaller than the space of the streaming algorithm, but a black-box application of Nisan's generator does not quite live up to this ideal: the seed length needed is larger than the space of the streaming algorithm by a multiplicative logarithmic factor. Furthermore, retrieving a block of the string output by the generator requires time proportional to the length of the seed, introducing a significant slowdown in many settings. Hence, obtaining streaming algorithms that are simultaneously space optimal and have a very fast update time—the time required to process each update in the stream—is challenging.

From Chapter 1, recall that in a turnstile stream, a vector $x \in \mathbb{R}^d$ is initially set to 0^d and receives updates of the form $(i_1, v_1), (i_2, v_2), \ldots, (i_m, v_m) \in [d] \times \{-M, \ldots, M\}$. On receiving an update of the form (i_j, v_j), the vector x is updated as follows: $x_{i_j} \leftarrow x_{i_j} + v_j$. Given a function f with domain $\mathbb{R}^d$, at the end of the stream we want to output an approximation to $f(x)$ using space sublinear in d while processing the stream. Some examples of f are (i) the F_p moments $\sum_{i=1}^{d} |x_i|^p$ and (ii) the number of distinct elements in the stream, often denoted by $\|x\|_0$. Turnstile streaming algorithms typically apply a randomized linear map $S : \mathbb{R}^d \to \mathbb{R}^D$ to the vector x and show that Sx can be used to approximate $f(x)$ at the end of processing the stream. The advantage of S being a linear map (or linear sketch) is that on receiving an update (i_j, v_j) in the stream, the sketch Sx can be updated by simply adding $(S)_{*i_j} v_j$ to the current sketch. Here $(S)_{*i_j}$ denotes the i_j-th column of the matrix S. Note that to obtain sublinear space algorithms, we cannot store the full matrix S in memory. One of the techniques here is to describe the entries of the matrix S using hash functions that are k-wise independent for a small value of k. For example, the CountSketch matrix of [CCF04] can be described by using 4-wise independent hash functions and thus can be stored efficiently. Further, for any $j \in [d]$, the column $(S)_{*j}$ can be generated efficiently using the hash functions, thereby allowing for a fast update of the sketch in the stream.

Unfortunately, it is not always easy to show that a matrix S generated using hash functions sampled from hash families with limited independence is sufficient to approximate $f(x)$. Indyk [Ind06] showed that we can assume full independence when constructing S and later derandomize[1] the construction of S using pseudorandom generators for small-space computation. The crucial idea of Indyk is that the final state of a linear sketch depends only on the vector x at the end of the stream and not on the sequence of updates that result in the vector x. Thus, any algorithm that assumes the columns of S are sampled independently can be derandomized using a pseudorandom generator that fools small space algorithms as follows: suppose an algorithm needs r uniform random bits to sample a column of S. Fixing a vector $x \in \mathbb{R}^d$, we construct a small space algorithm that makes a single pass over an $r \cdot d$ length uniform random string reading r uniform random bits at a time, sampling the column S_{*j}, and updating the stored sketch by adding $S_{*j}x_j$. If each coordinate of Sx can be stored using t bits, such an algorithm only uses $D \cdot t$ bits of space. As the columns of S were sampled independently, we use the analysis assuming full independence to conclude that the sketch computed by the algorithm has certain desired properties with a certain probability. Now, the small space algorithm can be *fooled*[2] using Nisan's PRG with a seed length of $\Theta(Dt \log(rd/Dt))$, which is $\Theta(Dt \log d)$ in many cases as Dt is often much smaller than d. The argument essentially shows that sampling columns of S using blocks of bits in the pseudorandom string is sufficient to ensure that the properties of Sx, which were proved assuming independent sampling of columns, still hold.

To implement the derandomized algorithm in a stream, given an update (i_j, v_j), we need to generate the column S_{*i_j} on the fly. Nisan's PRG allows us to generate a block of bits of the pseudorandom string by sequentially evaluating $O(\log d)$ hash functions $h : \{0,1\}^{Dt} \to \{0,1\}^{Dt}$ on a Dt length random seed. Thus, if the hash functions $h : \{0,1\}^{Dt} \to \{0,1\}^{Dt}$ used by the generator can be evaluated in time T on any input, then the block of bits necessary to generate the column S_{*i_j} can be computed in time $O(T \log d)$. We think of the pseudorandom string as a "hash function" mapping an index $i \in [d]$ to the block of pseudorandom bits needed to generate the column S_{*i}.

[1]We use the term "derandomize" to denote any procedure that *lowers* the randomness required by the algorithm for example by replacing a uniform random string with a string sampled from a Pseudorandom generator that uses a smaller uniform random seed.

[2]Formally, we say an algorithm using fully-random bits is fooled by a Pseudo Random Generator if the total variation distance between the distribution of outputs of the algorithm when using a string of fully random bits and when using a string sampled from the PRG is small.

The above argument of Indyk gives a *black box* way to derandomize a streaming algorithm albeit with a logarithmic space blowup (Dt bits to $Dt \log d$ bits) and an update time of $O(T \log d)$. We use the standard Word RAM model with a word size of $w = \Omega(\log d)$ bits to measure the time complexity of the algorithm. If $Dt \gg \log d$, then the hash functions $h : \{0,1\}^{Dt} \to \{0,1\}^{Dt}$ are slow to compute, making the update time $O(T \log d)$ very large. We will mostly express space usage of algorithms in *bits*, but with some Word RAM upper bounds having space measured in *words*.

Thus, a naïve application of Nisan's PRG results in a suboptimal algorithm space-wise, as well as a large update time. As numerous data stream algorithms use Nisan's PRG for derandomization, it is important to improve upon this. We significantly improve on the space overhead and update time for derandomization with our construction HashPRG. First, we show that by a careful analysis, in many problems, the pseudorandom string only has to fool an $O(\log d)$ space algorithm instead of the full $O(Dt)$ space algorithm. We show that such a reduction can be performed for the F_p moment estimation algorithms both for $p \in (0,2)$ and for $p > 2$. We further show that using a symmetry property of HashPRG, we can reduce the derandomization of CountSketch to fooling an $O(\log d)$ space algorithm. We note that CountSketch in our context cannot be derandomized by an application of Nisan's PRG to fooling an $O(\log d)$ space algorithm; see below for further discussion. We, however, are able to give a reduction to fooling an $O(\log d)$ space algorithm which enjoys two properties (i) the space complexity of the derandomized algorithm will be $O(Dt + \log^2 d)$, which is $O(Dt)$ when $Dt = \Omega(\log^2(d))$ and (ii) the hash functions that are to be evaluated to generate a block of pseudorandom bits that correspond to a column of $\boldsymbol{S}$ will now map the domain $\{0,1\}^{O(\log d)}$ to a range $\{0,1\}^{O(\log d)}$. Such hash functions can be evaluated in $O(1)$ time in the Word RAM model.

Even with our reduction to a PRG needing to fool only an $O(\log d)$ space algorithm, the need to evaluate $O(\log d)$ hash functions one after another to compute a block of pseudorandom bits presents a barrier to obtaining fast update time. We show that in the case of $Dt = d^{\Omega(1)}$, the space-vs-time trade-off of HashPRG lets us trade seed length for fast update time, by varying the number of hash functions needed in order to compute a block of pseudorandom bits.

10.1.1 Our Results

We construct a new pseudorandom generator, which we call HashPRG, that satisfies the same guarantees as Nisan's PRG but with an additional symmetry property. Our construction also allows a space-vs-time trade-off and lets us compute any block of pseudorandom bits quickly if we increase the seed length.

Theorem 10.1.1 (Informal). *There is a constant $c > 0$ such that for any positive integers n, b and k satisfying $b^k \le 2^{cn}$, there exists a pseudorandom generator parameterized by n, b and k that converts a random seed of length $O(bkn)$ bits to a bitstring of length $b^k \cdot n$ that cannot be distinguished from truly random bits by any Space(cn) algorithm making a single pass over the length $b^k \cdot n$ bitstring. A given n-bit block of this generator can be computed by evaluating k 2-wise independent hash functions mapping $\{0,1\}^n$ to $\{0,1\}^n$.*

Our generator uses the random seed of length $O(bkn)$ to sample $b \cdot k$ hash functions from a 2-wise independent hash family $\mathcal{H} = \{ h : \{0,1\}^n \to \{0,1\}^n \}$ such as [Die96, Theorem 3(b)] and uses n bits as an additional random seed.

In the Word RAM model with a word size $\Omega(n)$, an arbitrary block of n bits in the pseudorandom string generated by our generator can be computed in time $O(k)$. Fixing a value of $b^k = t$, we obtain that HashPRG needs a seed of size $O(t^{1/k}kn)$ to be able to generate a pseudorandom string of length $t \cdot n$ supporting the computation of an arbitrary block of pseudorandom bits in time $O(k)$. By varying k, we get a space versus time trade off. Even more informally, our result can be stated as follows:

Theorem 10.1.2 (Informal, Compare with [Nis92, Theorem 1]). *A space S algorithm making a single pass over a length $R \le \exp(S)$ random string can be fooled by a pseudorandom generator with a seed length $O((R/S)^{1/k} \cdot k \cdot S)$ where k is an integer parameter of HashPRG. A block of S random bits in the pseudorandom string can be computed by sequentially evaluating k hash functions mapping $\{0,1\}^S$ to $\{0,1\}^S$.*

Setting $k = \log(R/S)$ in the above theorem, we recover Nisan's result.

Another nice property of HashPRG is that the distribution of the pseudorandom string is symmetric in the following specific way. Let $\boldsymbol{\gamma}$ be sampled from HashPRG. Let $\boldsymbol{\gamma}$ be written as $\boldsymbol{\gamma}_0 \circ \boldsymbol{\gamma}_1 \circ \cdots \circ \boldsymbol{\gamma}_{b^k-1}$ where $\circ$ denotes concatenation and each $\boldsymbol{\gamma}_i$ is a length-n block. For arbitrary $\ell \in \{0, \ldots, b^k - 1\}$, define $\boldsymbol{\gamma}^{\oplus \ell} \coloneqq \boldsymbol{\gamma}_{0 \oplus \ell} \circ \boldsymbol{\gamma}_{1 \oplus \ell} \circ \cdots \circ \boldsymbol{\gamma}_{(b^k-1)\oplus \ell}$ where $\oplus$ denotes the bitwise xor operation. The construction of HashPRG ensures that $\boldsymbol{\gamma}^{\oplus \ell}$ has the same distribution as $\boldsymbol{\gamma}$. In all of our algorithms, we use a block of $\boldsymbol{\gamma}$ to generate appropriate random variables for the corresponding coordinate of the vector x that is being streamed. To analyze the properties of our streaming algorithms, we create *abstract algorithms*[3] that make a single pass over the pseudorandom string such that the behavior of the streaming algorithm is the same as the abstract algorithm. Since the distribution of the string $\boldsymbol{\gamma}$ is the same as the distribution of $\boldsymbol{\gamma}^{\oplus \ell}$, the distribution of outputs of the abstract algorithm when run on $\boldsymbol{\gamma}^{\oplus \ell}$ is the same as the distribution of the outputs when run directly on $\boldsymbol{\gamma}$. So given a fixed ℓ, the abstract algorithm can "know", throughout its execution, the block of pseudorandom bits corresponding to the coordinate ℓ that our original streaming algorithm uses. "Knowing" these bits is important in our analysis of the CountSketch data structure.

Applications. We use HashPRG to obtain space-optimal algorithms for constant factor F_p $(p > 2)$ estimation algorithms with an $O(1)$ update time in turnstile streams. Recall the turnstile stream setting: a vector $x \in \mathbb{R}^d$ is being maintained in the stream. The vector x is initialized to 0^d and receives a stream of updates $(i_1, v_1), (i_2, v_2), \ldots, (i_m, v_m) \in [d] \times \{-M, \ldots, M\}$. Unless otherwise specified, we assume in all of our results that $m, M \le \mathrm{poly}(d)$. For $0 < p < \infty$, we define $F_p(x) \coloneqq \sum_{i=1}^d |x_i|^p$ and $\ell_p(x) \coloneqq (\sum_{i=1}^d |x_i|^p)^{1/p}$.

Theorem 10.1.3. *Given $p > 2$, there is a turnstile streaming algorithm that uses $O(d^{1-2/p} \log d)$ words of space and outputs a constant factor approximation to $\|x\|_p$. Further, the streaming algorithm processes each update to the stream in $O(1)$ time in the Word RAM model on a machine with $\Omega(\log d)$ word size.*

[3]Algorithms that are created only for analysis and are not implemented.

We also show that for $p > 2$, similar techniques used in obtaining the above theorem can be used to obtain an algorithm that performs a relaxed version of the approximate ℓ_p sampling in the stream. See Section 10.4.4.

We next give an algorithm to estimate F_p moments for $0 < p < 2$ in the high accuracy regime. We show that the algorithm of [KNPW11] can be implemented using HashPRG *without a space blowup* and also ensure that the algorithm has a faster update time.

Theorem 10.1.4. *Let $0 < p < 2$ be a parameter and $1/\sqrt{d} \le \varepsilon \le 1/d^c$ be the desired accuracy for a constant $0 < c \le 1/2$. There is a streaming algorithm that uses $O(\varepsilon^{-2})$ words of space and outputs a $1 \pm \varepsilon$ approximation for $\|x\|_p^p$. The streaming algorithm processes each update to the stream in $O(\log d)$ time in the Word RAM model on a machine with $\Omega(\log d)$ word size.*

Using the reduction in [KNPW11] (Appendix A of the conference version), we can obtain a similarly improved update time for both additive and multiplicative entropy approximation in a stream.

We then derandomize the tighter analysis of CountSketch that [MP14] show assuming fully random hash functions. We prove that such tighter guarantees can be obtained even if the hash functions and sign functions in the CountSketch are generated from the pseudorandom string sampled using HashPRG, improving on the black-box derandomization using Nisan's generator.

Theorem 10.1.5 (Informal). *Given a table size t, number r of repetitions, and word size $w = \Omega(\log d)$, there exists a derandomization of CountSketch, $\mathsf{CS}_{HashPRG} : \{-M, \ldots, M\}^d \to \{-2^w, \ldots, 2^w\}^{tr}$, with the following properties:*

1. *The space complexity of the data structure is $O(tr + \log d)$ words.*
2. *Given an update (i, v), the data structure can be updated in time $O(r \log d)$ in the Word RAM model.*
3. *For any $\alpha \in [0, 1]$, given any index ℓ, an estimate of x_l given by $\hat{x}_\ell$ can be constructed from $\mathsf{CS}_{HashPRG}(x)$ such that*

$$\mathbf{Pr}[|x_\ell - \hat{x}_\ell| \ge \alpha\Delta] < 2\exp(-\alpha^2 r) + 2^{-Cw}$$

for $\Delta = \|tail_t(x)\|_2/\sqrt{t}$, where $tail_t(x) \in \mathbb{R}^d$ denotes the vector obtained after zeroing out the t entries with the highest absolute value in x.

We crucially use the symmetry property of HashPRG to obtain the above result. Note that when $t \cdot r \ge \log d$, the derandomization presents no asymptotic space blowup and retains the tighter analysis of estimation errors from [MP14]. Further, depending on the parameters t, r it is possible to obtain faster update time using the time-vs-space trade-off offered by HashPRG. We similarly derandomize the utility analysis of Private CountSketch from [PT22] to obtain:

Theorem 10.1.6 (Informal). *Consider private CountSketch, $\mathsf{PCS}(x) = \mathsf{CS}_{HashPRG}(x) + \boldsymbol{\nu}$ where the random variable $\boldsymbol{\nu} \sim N(0, \sigma^2)^D$, with r repetitions and table size t, where HashPRG uses block size w. Given $\ell \in [d]$ we can compute an estimate $\hat{x}_\ell$ from $\mathsf{PCS}(x)$ such that for every $\alpha \in [0, 1]$ and $\Delta = \|tail_t(x)\|_2/\sqrt{t}$,*

$$\mathbf{Pr}[|\hat{x}_\ell - x_\ell| \ge \alpha \max(\Delta, \sigma)] \le 2\exp(-\Omega(\alpha^2 r)) + O(2^{-cw}) \ .$$

We show the following tight result for estimating $\|x\|_\infty$ in a turnstile stream.

Theorem 10.1.7 (Informal). *Let x be an arbitrary d dimensional vector being maintained in a turnstile stream. Assuming that the coordinates of x are integers bounded in absolute value by* $\mathrm{poly}(d)$*, any streaming algorithm that estimates $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$ for $\varepsilon > ((\log d)/d)^{1/4}$ must use a space of $\Omega(\varepsilon^{-2} \log d \log 1/\varepsilon)$ bits. Matching this lower bound, there is an algorithm that uses $O(\varepsilon^{-2} \log d \log 1/\varepsilon)$ bits and outputs an approximation to $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$. The update time of this algorithm is $O(\log 1/\varepsilon)$ in the Word RAM model with a word size $\Omega(\log d)$.*

Our upper bound beats the previous best result of [BGW20, Theorem 10]. Their algorithm uses $O(\varepsilon^{-2} \log d(\log\log d + \log 1/\varepsilon))$ bits of space. Our matching lower bound shows that our result cannot be improved without making additional assumptions on the vector x.

When $\|x\|_\infty = \Theta(\|x\|_2)$, we show that it is possible to break the lower bound in the above result by giving an algorithm that uses $O(\varepsilon^{-2} \log d)$ bits of space and estimates $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$. The algorithm uses our derandomization of CountSketch with tight guarantees given in [MP14].

Theorem 10.1.8 (Informal). *Given a d dimensional vector x being updated in a turnstile stream, if $\|x\|_\infty = \Theta(\|x\|_2)$ there is a streaming algorithm that uses $O(\varepsilon^{-2} \log d)$ bits and approximates $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$ with probability $\geq 9/10$. The update time of this algorithm is $O(\log 1/\varepsilon)$ in the Word RAM model with a word size $\Omega(\log d)$.*

10.1.2 Previous Work

F_p estimation for $p > 2$. The problem of estimating moments in a stream has been heavily studied in the streaming literature and has been a source of lot of techniques both from the algorithms and lower bounds perspective. A lower bound of $\Omega(d^{1-2/p})$ bits on the space complexity of a constant factor approximation algorithm was shown in [CKS03, BYJKS04, Jay09]. On the algorithms side, [AKO11] gives a sketching algorithm with $m = O(d^{1-2/p} \log d)$ rows using a technique called *precision sampling*; this improves additional polylogarithmic factors of earlier work [IW05, BGKS06]. For linear sketches, [ANPW13] shows a lower bound of $m = \Omega(d^{1-2/p} \log d)$ on the number of rows in the sketch, thereby proving that the algorithm of [AKO11] is tight up to constant factors for linear sketching algorithms. All upper and lower bounds mentioned here are for constant factor approximation algorithms. See [And17, AKO11] and references therein for the upper and lower bounds for $(1 \pm \varepsilon)$-approximate algorithms for F_p estimation.

Later, Andoni [And17] gave a simpler linear sketch using $m = O(d^{1-2/p} \log d)$ rows for F_p moment estimation. Andoni uses the min-stability property of exponential random variables to embed ℓ_p into ℓ_∞ and then uses a CountSketch data structure to estimate the maximum absolute value in the vector obtained by sketching x with scaled exponential random variables. The analysis assumes that the exponential random variables are sampled independently. To derandomize the algorithm, Andoni uses the pseudorandom generator of Nisan and Zuckerman [NZ96] which shows that any randomized algorithm that uses space s and $\text{poly}(s)$ random bits can be simulated using $O(s)$ random bits. Since the space complexity of Andoni's algorithm is $d^{\Omega(1)}$, it can be derandomized with at most a constant factor blowup in space complexity. A major drawback of using the pseudorandom generator of Nisan and Zuckerman is that the update time of the sketch in the stream is $d^{\Omega(1)}$, which is prohibitively large. In this work, we show that we can derandomize Andoni's algorithm while having an update time of $O(1)$ in the Word RAM model.

We note that there are many other algorithms for F_p-moment estimation, such as [IW05, BGKS06, BO10, MW10, Gan15, GW18], which are based on subsampling the input vector in $O(\log d)$ scales and running an ℓ_2-heavy hitters algorithm at each scale. Although it may be possible to amortize the update time of the $O(\log d)$ levels of subsampling, such algorithms cannot achieve an optimal $O(1)$ update time since all known algorithms for ℓ_2-heavy hitters in the turnstile streaming model require $\Omega(\log d)$ update time.

F_p estimation for $p < 2$. Indyk [Ind06] showed how to estimate F_p moments for $p \in (0, 2]$ up to a factor $1 \pm \varepsilon$ in turnstile streams using a space of $O(\varepsilon^{-2} \log d)$ words, which translates to $O(\varepsilon^{-2} \log^2(d))$ bits with our assumption on the values of m, M. This work used *p-stable distributions* and as discussed earlier, introduced the influential technique to derandomize streaming algorithms using pseudorandom generators. Li [Li08] used p-stable distributions to define the geometric mean estimator, which can be used to give an unbiased estimator for $\|x\|_p^p$ with low variance and such that the mean of $\Theta(\varepsilon^{-2})$ independent copies of the estimator gives a $1 \pm \varepsilon$ estimate for $\|x\|_p^p$. Li's algorithm can also be derandomized to use $O(\varepsilon^{-2} \log^2 d)$ bits.

The upper bound was then improved to $O(\varepsilon^{-2} \log d)$ bits by Kane, Nelson and Woodruff [KNW10]. They avoid the $O(\log d)$ factor blowup which is caused when derandomizing using Nisan's PRG by showing that Indyk's algorithm can be derandomized using k-wise independent random variables for a small value of k. They also mention that a "more prudent analysis" of the seed length required in Nisan's generator to derandomize Indyk's algorithm makes the space complexity $O(\varepsilon^{-2} \log d + (\log d)^2)$ bits. The idea there was to use Nisan to fool a median of dot products, but it was not able to exploit fast update time, as we do for $0 < p < 2$, without our large alphabet improvement to Nisan's PRG.

They also show that any algorithm that $1 \pm \varepsilon$ approximates $\|x\|_p^p$ in a turnstile stream must use $\Omega(\varepsilon^{-2} \log d)$ bits of space, hence resolving the space complexity of F_p moment estimation in turnstile streams. Although their algorithm uses an optimal amount of space, the update time of their algorithm is $\widetilde{O}(\varepsilon^{-2})$ which is non-ideal when ε is small. Concurrent to their work, [AKO11] gave a streaming algorithm that has a fast update time of $O(\log d)$ per stream element but uses a suboptimal space of $O(\varepsilon^{-2-p} \log^2 d)$ bits for $p \in [1, 2]$.

[KNPW11] then made progress by giving algorithms that are space-optimal while having a fast update time. They give an algorithm that uses $O(\varepsilon^{-2} \log d)$ bits of space and with an update time of $O(\log^2(1/\varepsilon) \log\log(1/\varepsilon))$ per stream element. They use multiple techniques such as estimating the contribution of heavy and light elements to F_p separately by using new data structures and hash functions drawn from limited independent hash families; they buffer updates and use fast multipoint evaluation of polynomials and amortize the time over multiple updates to obtain fast update times.

When $\varepsilon < 1/d^c$ for a small enough constant c, which is when the $\widetilde{O}(\varepsilon^{-2})$ update time of earlier algorithms becomes prohibitive, we show that we can derandomize the algorithm of [KNPW11] using HashPRG. We show that there is a streaming algorithm using an optimal $O(\varepsilon^{-2}\log d)$ bits of space with an update time of $O(\log d)$ per stream element. Our algorithm updates the sketch immediately, removing the need to buffer updates and the use of fast multipoint polynomial evaluation from their algorithm. Our update time thus improves the previous update time of [KNPW11] from $O(\log^2 d \log\log d)$ to $O(\log d)$ for any polynomially small ε, making the first progress on this problem in over 10 years.

Estimation Error bounds with CountSketch. The CountSketch data structure [CCF04] can be used to compute an estimate $\hat{x}_\ell$ of x_ℓ for each $\ell \in [d]$. In [CCF04], the authors show that with high probability the maximum estimation error $\|x - \hat{x}\|_\infty \le \Delta$ where $\Delta = \|\text{tail}_k(x)\|_2/\sqrt{k}$ if the table size $t = O(k)$ and the number of repetitions $r = O(\log d)$. Minton and Price [MP14] observed that even though the worst-case estimation error follows the above law, most coordinates in $\hat{x}$ have asymptotically smaller estimation error. Concretely, they show that for any $\alpha \in [0, 1]$ and any coordinate $\ell \in [d]$, $\mathbf{Pr}[|\hat{x}_\ell - x_\ell| \ge \alpha\Delta] \le O(\exp(-\alpha^2 r))$.

They also show other applications of the above tighter analysis. While proving the above result, they assume that the sign functions used to construct the CountSketch data structure are fully random. They argue that their construction can be derandomized by incurring a factor $O(\log d)$ overhead in space using Nisan's PRG via the *black box* approach we described earlier. We derandomize CountSketch using HashPRG and show that even for the derandomized CountSketch construction, the above estimation error holds, albeit with an additional additive term from the failure of the pseudorandom generator. Our derandomized CountSketch data structure uses $O(r \cdot t + \log d)$ words of space. Note that $O(r \cdot t)$ words of space is anyway required to store the sketched vector and therefore when $\log d = O(r \cdot t)$, our derandomization increases the storage cost by at most a constant factor. Our derandomized CountSketch data structure has an update time of $O(r \log d)$ per stream element. We crucially use the symmetry property of HashPRG to reduce the problem to derandomizing a small space algorithm. While Minton and Price say that a derandomization of their algorithm with Nisan's PRG incurs an $O(\log d)$ factor space blow-up in the size of the CountSketch data structure, we observe that we can avoid the blow-up by a more careful analysis of derandomization using Nisan's PRG. We use the fact that Nisan's PRG is resilient to multiple passes [DPS11] as well to obtain the derandomization. See Section 10.6.2 for a discussion on how to derandomize CountSketch using Nisan's PRG and how the derandomization with Nisan's PRG compares against derandomization with HashPRG.

In the case of $rt = o(\log d)$, our derandomization is not ideal as the asymptotic space complexity of derandomized CountSketch is larger than a constant factor as compared to $O(r \cdot t)$ words of space. Jayaram and Woodruff [JW18] give an alternate derandomization of CountSketch with strong estimation error guarantees and show that their derandomized CountSketch needs $O(r \cdot t(\log\log d)^2)$ words of space. When $r \cdot t = o(\log d/(\log\log d)^2)$, their derandomization has a smaller space complexity than ours. They use a half-space fooling pseudorandom generator of Gopalan, Kane and Meka [GKM18] to derandomize CountSketch. However, the CountSketch data structure derandomized in [JW18] is a slight modification of the standard CountSketch data structure, and combined with the pseudorandom generator of [GKM18], leads to worse update times as compared to our derandomization of the standard CountSketch data structure. We also stress that in a number of applications of CountSketch, such as to the ℓ_2-heavy hitters problem, one has $r \cdot t = \Omega(\log d)$, and in this regime our derandomization is space-optimal, and not only significantly improves the update time, but also removes the additional $(\log\log d)^2$ factors in the space of [JW18].

Estimating $\|x\|_\infty$. Given a d dimensional vector x being maintained in a turnstile stream, it can be shown that approximating $\|x\|_\infty$ up to a multiplicative factor $C < \sqrt{2}$ requires $\Omega(d)$ bits of space by reducing from the INDEX problem. Notably, the lower bound for multiplicative approximation holds even in the stronger addition only model. Hence, the problem of approximating $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$ has gained more interest. Cormode, in the 2006 IITK workshop on data streams[4], asked if it was possible to approximate $\|x\|_\infty$ to an additive error of $\varepsilon\|x\|_2$ using fewer than $O(\varepsilon^{-2}\log^2 d)$ bits of space. Later [BCIW16] for insertion only streams and [BGW20] for general turnstile streams answered the question in affirmative by giving an algorithm that uses only $O(\varepsilon^{-2}\log d \log\log d)$ bits of space.

10.1.3 Technical Overview

HashPRG. We will briefly describe our construction and show how the construction and analysis differs from Nisan's. Given parameters n, b, k, our construction samples $b \cdot k$ independent hash functions $\boldsymbol{h}_i^{(j)} : \{0,1\}^n \to \{0,1\}^n$ for $(i,j) \in \{0,\ldots,k-1\} \times \{0,1,\ldots,b-1\}$ from a 2-wise independent hash family. We then sample a uniform random string $\boldsymbol{r} \sim \{0,1\}^n$, then the $b^k \cdot n$ pseudorandom string output by the generator is defined by $G_k(\boldsymbol{r}, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)$, where for any x,

$$G_0(x) \coloneqq x$$
$$G_k(x, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1}) \coloneqq G_{k-1}(\boldsymbol{h}_{k-1}^{(0)}(x), \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}) \circ \cdots \circ G_{k-1}(\boldsymbol{h}_{k-1}^{(b-1)}(x), \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{k-1}).$$

[4]Follow this link for the list of open problems.

Thus, for $i \in \{0, \dots, b^k - 1\}$ if i is written as $(i_{k-1} i_{k-2} \cdots i_0)$ in base b, then the i-th block of n bits in the pseudorandom string $G_k(\boldsymbol{r}, \boldsymbol{h}_1, \dots, \boldsymbol{h}_k)$ is given by

$$\boldsymbol{h}_0^{(i_0)}(\cdots(\boldsymbol{h}_{k-1}^{(i_{k-1})}(\boldsymbol{r}))).$$

Our analysis of HashPRG is based on a new, simpler, and more precise analysis of Nisan's generator [Nis92] (see Appendix C.1 for the definition of Nisan's PRG). Note that Nisan's generator corresponds to the special case of our generator where $b = 2$ and where for all i, x, we deterministically fix $\boldsymbol{h}_i^{(0)}(x) \coloneqq x$. With $\boldsymbol{h}_i^{(0)}$ the identity function, Nisan only defines a single hash function $h_i = \boldsymbol{h}_i^{(1)}$ for each i.

We will now pinpoint where our analysis diverges from a more natural generalization of Nisan's [Nis92]. The first main difference is in Definition 10.3.4. It plays a role similar to Nisan's

Definition 3. Let $A \subset \{0,1\}^n$, $B \subset \{0,1\}^m$, $h : \{0,1\}^n \longrightarrow \{0,1\}^m$, and $\varepsilon > 0$. We say that h is (ε, A, B)-independent if $|\,\mathbf{Pr}_{x \in \{0,1\}^n}[x \in A \text{ and } h(x) \in B] - \varrho(A)\varrho(B)| \le \varepsilon$, where $\varrho(A) = |A|/2^n$ and $\varrho(B) = |B|/2^m$.

However, our Definition 10.3.4 instead generalizes the following definition:

Definition 3'. Let $A \subset \{0,1\}^{n+m}$, $h : \{0,1\}^n \longrightarrow \{0,1\}^m$, and $\varepsilon > 0$. We say that h is (ε, A)-independent if $|\,\mathbf{Pr}_{x \in \{0,1\}^n}[(x, h(x)) \in A] - \varrho(A)| \le \varepsilon$, where $\varrho(A) = |A|/2^{n+m}$.

It turns out that replacing Definition 3 with Definition 3' in Nisan's proof yields a slightly tighter result. In the proof of [Nis92, Lemma 2], in step 2, all triplets of state nodes i, l, j are considered, where i is a start state, l is an intermediate state and j is an end state. Referring to Definition 3, Nisan uses $A = B_{i,l}^{h_1,\dots,h_{k-1}} \subset \{0,1\}^n$ and $B = B_{l,j}^{h_1,\dots,h_{k-1}} \subset \{0,1\}^n$. If instead we base the analysis on Definition 3', then we only need to consider pairs of state nodes i, j and use $A = B_{i,j}^{h_1,\dots,h_{k-1}} \subset \{0,1\}^{2n}$. This affects the whole analysis of Nisan, but it turns out that it only gets simpler, and this was the starting point for our generalized analysis. Avoiding the intermediate state l was crucial for us because we would need $b-1$ intermediate states $l_1 \dots l_{b-1}$, and this would lead to much worse bounds with larger b.

Our construction also implies that the pseudorandom generator has a symmetry property. To see the symmetry, suppose we define $\boldsymbol{h}'_0, \dots, \boldsymbol{h}'_{k-1}$ with $(\boldsymbol{h}'_0)^{(0)} = \boldsymbol{h}_0^{(1)}$, $(\boldsymbol{h}'_0)^{(1)} = \boldsymbol{h}_0^{(0)}$ and $(\boldsymbol{h}'_i)^{(j)} = \boldsymbol{h}_i^{(j)}$ in all other cases. Then the string $G_k(\boldsymbol{r}, \boldsymbol{h}'_0, \dots, \boldsymbol{h}'_{k-1})$ is obtained by an appropriate permutation of blocks in the string $G_k(\boldsymbol{r}, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})$. As both the strings $G_k(\boldsymbol{r}, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})$ and $G_k(\boldsymbol{r}, \boldsymbol{h}'_0, \dots, \boldsymbol{h}'_{k-1})$ are just as likely when sampling from HashPRG, we obtain the symmetry property.

Extending the above switching argument, we get the following family of transformations that preserve the distribution of the pseudorandom string. Consider an arbitrary $\ell \in \{0, \dots, b^k - 1\}$. Let $\ell_{k-1} \cdots \ell_0$ be the base b representation of ℓ. Suppose $\boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1}$ be the hash functions sampled in the construction of HashPRG. Define $\boldsymbol{h}'_0, \ldots, \boldsymbol{h}'_{k-1}$ as

$$(\boldsymbol{h}'_i)^{(j)} \coloneqq \boldsymbol{h}_i^{(j \oplus \ell_i)}$$

for all $i \in \{ 0, \ldots, k-1 \}$ and $j \in \{ 0, \ldots, b-1 \}$. Clearly, the joint distribution of $(\boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})$ is the same as $(\boldsymbol{h}'_0, \ldots, \boldsymbol{h}'_{k-1})$. Now, for any fixed x, we have

$$G_k(x, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})^{\oplus \ell} = G_k(x, \boldsymbol{h}'_0, \ldots, \boldsymbol{h}'_{k-1}).$$

Hence, if $\boldsymbol{y} \sim$ HashPRG, then $\boldsymbol{y}^{\oplus \ell}$ has the same distribution as $\boldsymbol{y}$.

F_p estimation for $p > 2$. We derandomize Andoni's algorithm for F_p moment estimation [And17]. Andoni's algorithm can be seen as sketching in two stages: let $\boldsymbol{z} \in \mathbb{R}^d$ be a random vector such that $\boldsymbol{z}_i = \boldsymbol{E}_i^{-1/p} x_i$ where $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_d$ are independent standard exponential random variables. By the min-stability property of exponential random variables, we obtain that $\|\boldsymbol{z}\|_\infty^p \sim \|x\|_p^p / \boldsymbol{E}$, where $\boldsymbol{E}$ is also a standard exponential random variable. Thus, the coordinate of maximum absolute value in $\boldsymbol{z}$ can be used to estimate $\|x\|_p$. Notice that we have not yet performed any dimensionality reduction. Then a CountSketch matrix $\boldsymbol{S}$ with $O(d^{1-2/p} \log d)$ rows is constructed using $O(\log d)$-wise independent hash functions, and this is applied to the vector $\boldsymbol{z}$ to obtain $\boldsymbol{f} = \boldsymbol{S}\boldsymbol{z}$. Andoni argues using the properties of exponential random variables that all of the following hold true simultaneously with a large constant probability: (i) $\|\boldsymbol{z}\|_\infty$ is a constant factor approximation to $\|x\|_p$, (ii) there are only $O((\log d)^p)$ coordinates in $\boldsymbol{z}$ with absolute value greater than $\|x\|_p / c \log d$ and (iii) $\|\boldsymbol{z}\|_2^2 \le O(d^{1-2/p} \|\boldsymbol{x}\|_2^2)$. Conditioned on these properties of $\boldsymbol{z}$, Andoni argues that $\|\boldsymbol{f}\|_\infty \approx \|\boldsymbol{z}\|_\infty \approx \|x\|_p$ with a large probability. Hence, $\|\boldsymbol{f}\|_\infty$ is a constant factor approximation for $\|x\|_p$ with a large constant probability.

We only have to derandomize the exponential random variables as $\boldsymbol{S}$ is constructed using $O(\log d)$-wise independent hash functions which can be efficiently stored and evaluated [CPT15]. So, we want to show that each of the three properties of the vector $\boldsymbol{z}$ hold even when the exponential random variables are sampled using a pseudorandom string. Now fix a vector x. There is an $O(\log d)$ space algorithm that makes a single pass over the string used to generate exponential random variables and (1) computes $\|\boldsymbol{z}\|_\infty$, (2) computes the number of coordinates in $\boldsymbol{z}$ with absolute value at least $\|x\|_p / c \log d$, and (3) computes $\|\boldsymbol{z}\|_2^2$. The algorithm is simple: it goes over a block of the string to generate $\boldsymbol{E}_1$, sets $\|\boldsymbol{z}\|_\infty = |\boldsymbol{E}_1^{-1/p} x_1|$, increases a counter if $|\boldsymbol{E}_1^{-1/p} x_1| \ge \|x\|_p / (c \log d)$ and sets $\|\boldsymbol{z}\|_2^2 = (\boldsymbol{E}_1^{-1/p} x_1)^2$, and proceeds to read the next block of bits to generate $\boldsymbol{E}_2$ and update the variables using the value of $\boldsymbol{E}_2$ accordingly and continues so on. Now, using any PRG that fools an $O(\log d)$ space algorithm, we obtain that the random variables $\boldsymbol{E}_i$ constructed using the pseudorandom string also make the vector $\boldsymbol{z}$ have each of the three properties. Using the time-vs-space trade-off of HashPRG, we obtain that any block of the pseudorandom string can be obtained in $O(1)$ time if the seed length of HashPRG is d^ε for a small constant $\varepsilon > 0$, thus making the time to compute a block of pseudorandom bits $O(1)$ in the Word RAM model. Further, the $O(\log d)$-wise independent hash functions necessary to map the vector $\boldsymbol{z}$ to $\boldsymbol{f}$ when sampled from the constructions of [CPT15] allow for the hash functions to be evaluated in $O(1)$ time. The data structures that allow the hash value to be computed in $O(1)$ time can be stored in d^{ε} bits of space as well for any constant $\varepsilon > 0$. Thus, the overall update time of the algorithm is $O(1)$ in the Word RAM model. If ε is chosen smaller than $1 - 2/p$, then the asymptotic space complexity of our derandomized F_p moment estimation algorithm remains $O(d^{1-2/p} \log d)$ words, while having a very fast $O(1)$ update time.

We note that the F_p estimation problem for $p > 2$ is ideally suited for HashPRG — it is precisely because the algorithm uses a large amount of memory already that we are able to use our generator over a large alphabet without a space overhead, which then allows us to remove the $O(\log d)$ factor in the update time and achieve constant time. Moreover, it is critical that we can keep track of various quantities needed to fool the algorithm with only $O(1)$ words of memory, as this is also needed for fast update time in our derandomization.

F_p estimation for $p < 2$. We give an alternate derandomization of the algorithm of [KNPW11]. Their algorithm is based on Li's geometric mean estimator [Li08] using p-stable random variables. They introduce two new data structures they call HighEnd and LightEstimator. They also concurrently run an ℓ_p heavy hitters algorithm and at the end of processing the stream, they use the heavy hitters data structure to find a set $L \subseteq [d]$ of coordinates such that $\{i \mid |x_i|^p \geq (\alpha/2)\|x\|_p^p\} \supseteq L \supseteq \{i \mid |x_i|^p \geq \alpha\|x\|_p^p\}$. They then use the HighEnd data structure to estimate $\|x_L\|_p^p$ up to a factor of $1 \pm \varepsilon$. By definition of L, all the coordinates in $x_{[d]\setminus L}$ have a small magnitude. Using this fact, they show that their LightEstimator data structure can be used to give a low variance estimator for $\|x_{[d]\setminus L}\|_p^p$. By running multiple independent copies of LightEstimator concurrently, they obtain an accurate estimate for $\|x_{[d]\setminus L}\|_p^p$ and then output the sum of estimates of $\|x_L\|_p^p$ and $\|x_{[d]\setminus L}\|_p^p$.

The HighEnd estimator can be maintained using $O(\varepsilon^{-2} \log d)$ bits and has an update time of $O(\log d)$. Hence, we focus on giving a more efficient derandomization of LightEstimator. At a high level, they hash coordinates of x into $O(1/\alpha)$ buckets and for each bucket they maintain Li's estimator for the F_p moment of coordinates hashed into that bucket. At the end of the stream, the set L is revealed, and they output the sum of Li's estimators of the buckets into which none of the elements of L are hashed into. They scale the sum appropriately to obtain an unbiased estimator to $\|x_{[d]\setminus L}\|_p^p$. They show that hashing of the coordinates can be quickly performed using the hash family of [PP08]. The only thing that remains is to derandomize Li's estimator in each individual bucket. They prove that the p-stable random variables in Li's estimator can be derandomized by using $O(1/\varepsilon^p)$-wise independent random variables and show that this is sufficient to obtain algorithms with an optimal space complexity of $O(\varepsilon^{-2} \log d)$ bits and an update time of $O(\log^2(1/\varepsilon) \log\log(1/\varepsilon))$. While the other parts of their algorithm have fast update times, the LightEstimator derandomized using limited independent p-stable random variables leads to a slow update time. We give an alternate derandomization of LightEstimator using HashPRG.

Fix a particular bucket b and the hash function h that hashes the coordinates into one of the buckets. We give an $O(\log d)$ space algorithm that makes a single pass over the string used to generate p-stable random variables and computes Li's estimator for bucket b. The algorithm simply makes a pass over the string, and if a coordinate i gets hashed into bucket b, it uses the block of bits in the string corresponding to the i-th coordinate to generate p-stable random variables and updates Li's estimator for bucket b using the generated p-stable random variables. The existence of such an algorithm implies that the expectation of Li's estimator is fooled by HashPRG and therefore the expectation of the sum of Li's estimators over all the buckets is fooled as well by HashPRG. We now need to bound the variance of the sum of estimators of the b buckets. Here to fool the variance, that is, we only have to fool $\mathbf{E}[\mathbf{Est}_b \cdot \mathbf{Est}_{b'}]$ for pairs of buckets b, b'. The idea of using a PRG to fool the variance for a streaming problem has also been used in [CIW24]. Here $\mathbf{Est}_b$ denotes the result of Li's estimator for bucket b. Now, for any fixed pair of buckets b, b' we again have that there is an $O(\log d)$ space algorithm that computes Li's estimators for buckets b and b' simultaneously while making a single pass over the string used to generate p-stable random variables. This shows that HashPRG fools $\mathbf{E}[\mathbf{Est}_b \cdot \mathbf{Est}_{b'}]$ and hence the overall variance of the estimator. As we only need to fool the mean and variance, we obtain a derandomization of p-stable random variables using HashPRG. Additionally, when $\varepsilon < 1/d^c$, we can use the time-vs-space trade-off of HashPRG to obtain an update time of $O(\log d)$ without changing the asymptotic space complexity of the algorithm. Note that the case of ε being small is actually the setting for which we would most like to improve the update time of [KNPW11].

Entropy estimation. An improved update time for F_p moment estimation for $p \in (0, 2)$ also leads to improved update time for entropy estimation using the algorithm of [HNO08]. See Section A in the conference version of [KNPW11] for a discussion on how the update time of F_p moment estimation algorithms translate to update time of approximate entropy estimation algorithms.

CountSketch. CountSketch is a randomized linear map $\mathsf{CS} : \mathbb{R}^d \to \mathbb{R}^D$ defined by two parameters: (i) the table size t and (ii) the number of repetitions r. Here $D = r \cdot t$. For each $i \in [r]$, we have a hash function $\boldsymbol{g}_i : [d] \to [t]$ and a sign function $\boldsymbol{s}_i : [d] \to \{+1, -1\}$. Indexing the coordinates of $\mathsf{CS}(x)$ by $(i, j) \in [r] \times [t]$, we define $\mathsf{CS}(x)_{i,j} = \sum_{\ell \in [d]} [\boldsymbol{g}_i(\ell) = j] \boldsymbol{s}_i(\ell) x_\ell$. Thus, for each repetition i, the coordinate x_ℓ is multiplied with a sign $\boldsymbol{s}_i(\ell)$ and added to the $\boldsymbol{g}_i(\ell)$-th bucket. For each $\ell \in [d]$, we can define $\hat{x}_\ell = \text{median}(\{\boldsymbol{s}_i(\ell) \cdot \mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)} \mid i \in [r]\})$. The randomness in a CountSketch data structure is from the hash functions $\boldsymbol{g}_i$ and the sign functions $\boldsymbol{s}_i$. Assuming that the hash functions $\boldsymbol{h}_i$ and $\boldsymbol{s}_i$ for $i \in [t]$ are drawn independently from 2-wise and 4-wise independent hash families respectively, [CCF04] showed that if $r = O(\log d)$, then with probability at least $1 - 1/\text{poly}(d)$, $\|x - \hat{x}\|_\infty \le \Delta$ for $\Delta = \|\text{tail}_k(x)\|_2/\sqrt{k}$ if $t = O(k)$. As discussed in Section 10.1.2, Minton and Price assume that the hash functions $\boldsymbol{g}_i$ and the sign functions $\boldsymbol{s}_i$ are fully random to give probability bounds on estimation error for any particular index $\ell \in [d]$. We derandomize their construction by using a pseudorandom generator to sample the hash functions $\boldsymbol{g}_i$ and the sign functions $\boldsymbol{s}_i$. Suppose we treat a bitstring γ as $t \cdot d$ blocks of equal length. We index the blocks of γ by $(i, \ell) \in [r] \times [d]$. We use the block $\gamma_{i,\ell}$ to define $\boldsymbol{g}_i(\ell)$ and $\boldsymbol{s}_i(\ell)$ in the natural way. If γ is sampled uniformly at random, then clearly we have that $\boldsymbol{g}_i$ and $\boldsymbol{s}_i$ constructed using the string γ are fully random and the CountSketch data structure constructed using such hash functions satisfies the guarantees given by Minton and Price. We need to define which block of the string corresponds to which (i, ℓ), since when we receive an update in the stream, we need to be able to extract the corresponding block from the pseudorandom string. Now, we want to show that even if γ is sampled from HashPRG, we obtain similar guarantees on the estimation error. Fix a vector x and coordinate ℓ. Our strategy to derandomize has been to give an algorithm using space $O(\log d)$ bits that makes a single pass over the randomness and computes the quantity of interest. Now, in a black box way, we can conclude that the distribution of the quantity of interest does not change much even if a string sampled from HashPRG is used instead of a fully random string.

We shall try to employ the same strategy here. Fixing an $x \in \mathbb{R}^d$, $\ell \in [d]$ and $\alpha \in [0, 1]$, we want to walk over the string γ and count the number of repetitions $i \in [r]$ for which the value $\boldsymbol{s}_i(\ell) \cdot \mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)} > x_\ell + \alpha\Delta$ and the number of repetitions $i \in [r]$ for which the value $\boldsymbol{s}_i(\ell) \cdot \mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)} < x_\ell - \alpha\Delta$. Clearly the estimator $\hat{x}_\ell \in [x_\ell - \alpha\Delta, x_\ell + \alpha\Delta]$ if and only if both counts are smaller than $r/2$ (for odd r). Hence, if both the counts can be computed by a small space algorithm, we can derandomize CountSketch using HashPRG. We immediately hit a roadblock while trying to design such an algorithm. As we fixed an index $\ell \in [d]$, for repetition $i \in [r]$, the quantity of interest is $\mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)} \coloneqq \sum_{\ell' \in [d]} [\boldsymbol{g}_i(\ell') = \boldsymbol{g}_i(\ell)] \boldsymbol{s}_i(\ell') x_{\ell'}$. As the string γ is ordered in increasing order of ℓ' for each repetition i, the algorithm does not know the value of $\boldsymbol{g}_i(\ell)$ until it gets to the block $\gamma_{i,\ell}$. So, for indices $\ell' < \ell$, the algorithm is not aware if $\boldsymbol{g}_i(\ell')$ equals $\boldsymbol{g}_i(\ell)$ or not and hence cannot track the value of $\mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)}$ with a single pass over γ. To solve the problem, we use the symmetric property unique to HashPRG. As we mentioned in Section 10.1.1, if $\boldsymbol{\gamma} \sim$ HashPRG, then for any integer m, the string $\boldsymbol{\gamma}^{\boxplus m}$ has the same distribution as $\boldsymbol{\gamma}$. Making use of this symmetry property, we can now assume that the block $\boldsymbol{\gamma}_{i,j}$ actually corresponds to the hash values of the index $j \boxplus \ell$ for iteration i as the joint distribution of hash and sign values defined by the earlier ordering of the blocks will be the same as the new ordering of the blocks by the symmetry property.

Thus, reading the block $\boldsymbol{\gamma}_{i,1}$, the algorithm immediately knows which bucket the index ℓ gets hashed into in the i-th repetition of CountSketch. Now in a single pass over the blocks $\boldsymbol{\gamma}_{i,1}, \ldots, \boldsymbol{\gamma}_{i,d}$, an $O(\log d)$ space algorithm can compute $\boldsymbol{s}_i(\ell)\mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)}$ and at the end of traversing the blocks corresponding to the i-th iteration, the algorithm checks if $\boldsymbol{s}_i(\ell)\mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)}$ is $> x_\ell + \alpha\Delta$ or $< x_\ell - \alpha\Delta$ and updates the corresponding counters accordingly. Thus, we have an $O(\log d)$ space algorithm and HashPRG fools an $O(\log d)$ space algorithm and can be used to derandomize CountSketch with the stronger guarantees as given by Minton and Price without incurring a space blowup when $\log d = O(r \cdot t)$.

For each update in the stream, and for each of the r repetitions, we need to compute a block of the pseudorandom string which takes $O(\log d)$ time in the Word RAM model with a word size $w = \Omega(\log d)$. Hence, our derandomized CountSketch with strong guarantees from [MP14] has an update time of $O(r \log d)$. When $r \cdot t = d^{\Omega(1)}$, using the space-vs-time trade-off of HashPRG, we can obtain an update time of $O(r)$.

Private CountSketch. In a recent work, Pagh and Thorup [PT22] gave an improved analysis of the estimation error of differentially private CountSketch. After computing $\mathrm{CS}(x)$ in the stream, they compute $\mathrm{PCS}(x) = \mathrm{CS}(x) + \boldsymbol{\nu}$ where $\boldsymbol{\nu} \sim N(0, \sigma^2)^D$. They show that for an appropriate value of σ, $\mathrm{PCS}(x)$ is (ε, δ)-differentially private. They also show that for any $\ell \in [d]$, the estimator $\hat{x}_\ell$ computed using $\mathrm{PCS}(x)$ also concentrates heavily around x_ℓ, and gave similar concentration bounds to that of Minton and Price [MP14]. The analysis of [PT22] assumes that the hash functions and sign functions are fully random – the same assumption as in [MP14]. We derandomize the Private CountSketch construction using HashPRG, which is similar to our derandomization of [MP14].

Approximating $\|x\|_\infty$. We consider the problem of approximating $\|x\|_\infty$ up to an additive error $\varepsilon\|x\|_2$. Directly using the CountSketch data structure, we can approximate each coordinate of the vector x up to an additive error of $\varepsilon\|x\|_2$ using $O(\varepsilon^{-2}(\log d)^2)$ bits. In the space complexity, a $\log d$ factor comes from the fact that we have to store integers with magnitudes $\mathrm{poly}(d)$ and other $\log d$ factor is because of $O(\log d)$ repetitions of CountSketch to be able to take a union bound over the reconstruction error of all $\mathrm{poly}(d)$ coordinates. We show that there is a simple linear sketching technique to reduce the dimension from d to $\mathrm{poly}(1/\varepsilon)$ while preserving the $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$. Then the CountSketch data structure, using a space of $O(\varepsilon^{-2} \log 1/\varepsilon \log d)$ bits, can be used to approximate the $\|\cdot\|_\infty$ of the sketched vector thereby approximating $\|x\|_\infty$. We use the randomized map $\boldsymbol{L} : \mathbb{R}^d \to \mathbb{R}^t$ defined as follows to reduce the dimension: let $\boldsymbol{h} : [d] \to [t]$ be drawn at random from a 2-wise independent hash family and $\boldsymbol{s} : [d] \to \{+1, -1\}$ be drawn at random from a 4-wise independent hash family. Define $(\boldsymbol{L}x)_i = \sum_{j \in [d] : \boldsymbol{h}(i) = j} \boldsymbol{s}(j) x_j$. Using simple variance computations, we show that if $t = \mathrm{poly}(1/\varepsilon)$, with a high probability, $\|\boldsymbol{L}x\|_\infty = \|x\|_\infty \pm \varepsilon\|x\|_2$ and that $\|\boldsymbol{L}x\|_2 \le 2\|x\|_2$. Note that a turnstile update to one coordinate of x simply translates to a turnstile update to one coordinate of $\boldsymbol{L}x$ and hence the two stage sketching algorithm can be efficiently implemented in a stream. A similar universe reduction was employed in [KNPW11], but the technique was previously not explored in the context of ℓ_∞ estimation.

We further show that this simple algorithm is tight by showing an $\Omega(\varepsilon^{-2} \log 1/\varepsilon \log d)$ bits lower bound on any algorithm that approximates $\|x\|_\infty$ up to an additive $\varepsilon\|x\|_2$. We define a communication problem called "Augmented Sparse Set-Disjointness". In this one-way communication problem, Alice receives sets $A_1, \ldots, A_t$ with the property that $|A_i| = k$ and $A_i \subseteq [n]$ for all $i \in [t]$. Bob similarly receives the sets $B_1, \ldots, B_t$ with the same properties. Given an index $i \in [t]$ and the sets $A_1, \ldots, A_{i-1}$, using a single message M (possibly randomized) from Alice, Bob has to compute if $B_i \cap A_i = \emptyset$ or not. In the case of $t = 1$, [DKS10] show that if $n \ge k^2$, then the sparse set disjointness problem has a communication lower bound of $\Omega(k \log k)$. Note that for $t = 1$, simply sending the entire set A_1 to Bob requires a communication of $O(k \log n)$ bits. Surprisingly, they show that there is a protocol using $O(k \log k)$ bits to solve the problem. Extending their ideas, we show that the Augmented Sparse Set-Disjointness problem has a communication lower bound of $\Omega(tk \log k)$. Embedding an instance of the Sparse Set-Disjointness into approximating $\|x\|_\infty$ for an appropriate vector x, we show a lower bound of $\Omega(\varepsilon^{-2} \log 1/\varepsilon \log d)$ bits.

The hard distribution in the lower bound has the property that the vector x constructed satisfies $\|x\|_\infty = O(\varepsilon\|x\|_2)$. We show that it is possible to break the lower bound if we assume that $\|x\|_\infty \geq c\|x\|_2$ for a constant c. This is the case when the coordinates of x follow Zipf's law where the i-th largest coordinate has a value approximately $i^{-\alpha}$ for $\alpha > 0.5$. The algorithm is again the two stage sketch we described but in the second stage, instead of using CountSketch as described in [CCF04] using constant wise independent hash functions, we use the tighter CountSketch guarantees that we obtain by derandomizing the analysis of [MP14] using HashPRG. We show that after the first level sketching, only a few large coordinates need to be estimated to large accuracy while the rest of the coordinates can have larger estimation errors. Using this insight, we show that $O(\varepsilon^{-2} \log d)$ bits of space is sufficient to estimate $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$.

10.2 Preliminaries

Notation. For an integer $n \geq 1$, let $[n]$ denote $\{1, 2, \ldots, n\}$. For a predicate P let $[P]$ have the value 1 when P is true and the value 0 when P is false. Let $\mathrm{tail}_t(x)$ denote the vector derived from x by changing the t entries with the largest absolute value to zero. We use bold symbols such as $\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{M}, \ldots$ to denote that these objects are explicitly sampled from an appropriate distribution.

For a real-valued matrix M define $\|M\| = \sup_{x \neq 0} \|xM\|_1 / \|x\|_1$ where $\|x\|_1 = \sum_i |x_i|$. (This matrix norm is sometimes written $\|M\|_1$, but since we do not need other matrix norms we omit the subscript.)

Model of Computation. All of our running times are in the Word RAM model with a word size $O(\log d)$ unless otherwise mentioned. We assume that all elementary operations on words can be performed in $O(1)$ time.

Definition 10.2.1 (k-wise independence)**.** A family of hash functions $\mathcal{H} = \{ h : [u] \to [v] \}$ is said to be k-wise independent if for any k distinct keys $x_1, x_2, \ldots, x_k \in [u]$ and not necessarily distinct values $y_1, y_2, \ldots, y_k \in [v]$

$$\mathbf{Pr}_{\boldsymbol{h} \sim \mathcal{H}}[\boldsymbol{h}(x_1) = y_1 \wedge \cdots \wedge \boldsymbol{h}(x_k) = y_k] = \frac{1}{v^k}.$$

The definition states that if $\boldsymbol{h} \sim \mathcal{H}$, then for any $x \in [u]$, the random variable $\boldsymbol{h}(x)$ is uniformly distributed over $[v]$ and for any k distinct keys $x_1, \ldots, x_k \in [u]$, the random variables

$\boldsymbol{h}(x_1), \ldots, \boldsymbol{h}(x_k)$ are independent.

We now state a randomized construction of a hash family from [CPT15] that lets us evaluate the sampled hash function quickly on any input.

Theorem 10.2.2 (Corollary 3 in [CPT15]). *There exists a randomized data structure that takes as input positive integers $u, v = u^{O(1)}, t, k = u^{O(1/t)}$ and selects a family of function $\mathcal{F}$ from $[u]$ to $[v]$. In the Word RAM model with word length $\Theta(\log u)$ the data structure satisfies the following:*

1. *The space used to represent the family $\mathcal{F}$ as well as a function $f \in \mathcal{F}$ is $O(ku^{1/t}t)$ bits.*
2. *The evaluation time of any function $f \in \mathcal{F}$ on any input is $O(t \log t)$.*
3. *With probability $\geq 1 - u^{-1/t}$, we have that $\mathcal{F}$ is a k-wise independent family.*

Throughout the chapter, we use this construction with a constant t and k at most $O(\log u)$.

We now state the guarantees of the hash family construction from [PP08]. Whereas the above construction gives a randomized hash family $\mathcal{F}$ that is k-wise independent with some probability, the following construction gives a randomized hash family $\mathcal{H}$ that when restricted to any fixed subset S of a certain size is a uniform hash family with some probability. We use this construction when we want to ensure that all elements of a small underlying set (but unknown to the streaming algorithm) are hashed to uniformly random locations.

Theorem 10.2.3 (Theorem 1 of [PP08]). *Let $S \subseteq U = [u]$ be a set of $z > 1$ elements and let $V = [v]$ for any $1 \leq v \leq u$. Suppose the machine word size is $\Omega(\log u)$. For any constant $C > 0$, there is a Word RAM algorithm that, using $\log(z)(\log v)^{O(1)}$ time and $O(\log z + \log\log u)$ bits of space, selects a family of hash functions $\mathcal{H}$ from U to V (independent of S) such that*

- *$\mathcal{H}$ is z-wise independent (in other words, uniform) when restricted to S, with probability $1 - O(1/z^C)$.*
- *Any function $h \in \mathcal{H}$ can be represented using $O(z \log v)$ bits and h can can be evaluated on any $x \in U$ in $O(1)$ time in the Word RAM model. The data structure (or representation) of a random function from the family $\mathcal{H}$ can be constructed in $O(z)$ time.*

We now define ε pseudorandom generators for any given class of algorithms $\mathcal{C}$.

Definition 10.2.4. A generator $G : \{0,1\}^n \to \{0,1\}^m$ is called an ε pseudorandom generator for the class of algorithms $\mathcal{C}$ if for any $C \in \mathcal{C}$,

$$|\mathbf{Pr}_{\boldsymbol{x} \sim U_m}[C(\boldsymbol{x}) \text{ accepts}] - \mathbf{Pr}_{\boldsymbol{y} \sim U_n}[C(G(\boldsymbol{y})) \text{ accepts}]| < \varepsilon.$$

Here U_m denotes the uniform distribution over $\{0,1\}^m$. For a generator G, the quantity n is called the *seed length*. In this work $\mathcal{C}$ is taken to be the set of space bounded algorithms with an appropriate space parameter s. The algorithms have a read/write space of s bits and a read only tape that contains an input. The algorithms are allowed to only stream over the m length random/pseudorandom string. Although the above definition is in terms of accept/reject, it can be extended to general functions by instead considering the total variation distance between the distributions of the outputs.

10.3 HashPRG

In this section we present a new pseudorandom generator for space-bounded computations, which is going to be our main tool for derandomizing streaming algorithms. The starting point is the classical generator of Nisan [Nis92] (summarized in Appendix C.1), which we extend to provide a trade-off between seed length and the time to compute an arbitrary output block. To make our treatment easy to access for readers familiar with Nisan's generator, we will follow the same proof outline, but make crucial changes to avoid a union bound over all possible intermediate states. An advantage of our generator over Nisan's, even in the setting where the seed lengths are the same, is that it has a certain *symmetry* property that we need in our applications.

10.3.1 HashPRG Construction

Let $b \geq 2$ be an integer. Consider $\boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1}$ where $\boldsymbol{h}_i \coloneqq (\boldsymbol{h}_i^{(0)}, \dots, \boldsymbol{h}_i^{(b-1)})$ is a vector of b hash functions with each $\boldsymbol{h}_i^{(j)} : \{0,1\}^n \to \{0,1\}^n$ being a hash function drawn independently from a 2-wise independent hash family $\mathscr{H}$. We slightly abuse terminology and also refer to the vectors $\boldsymbol{h}_i$ as hash functions.

Using hash functions $\boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1}$, define a generator $G_k : \{0,1\}^n \to \{0,1\}^{n \cdot b^k}$ recursively:

$$\begin{aligned} G_0(x) &= x \\ G_k(x, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1}) &= G_{k-1}(\boldsymbol{h}_{k-1}^{(0)}(x), \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-2}) \circ \cdots \circ G_{k-1}(\boldsymbol{h}_{k-1}^{(b-1)}(x), \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-2}). \end{aligned}$$

Here $\circ$ denotes concatenation. Note that Nisan's generator can be obtained by setting $b = 2$ and deterministically setting $\boldsymbol{h}_i^{(0)}(x) \coloneqq x$ for all i, x.

By construction, we have, for $x \in \{0,1\}^n$, that $G_k(x, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})$ is a bitstring of length $n \cdot b^k$. We look at the string $G_k(x, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})$ as concatenation of b^k chunks each of length n, chunks indexed by $0, \dots, b^k - 1$. Let $j \in \{0, \dots, b^k - 1\}$ be written as $j_{k-1} \cdots j_0$ in base b. Then the jth chunk of the string $G_k(x, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})$ is given by

$$\boldsymbol{h}_0^{(j_0)}(\boldsymbol{h}_1^{(j_1)}(\cdots \boldsymbol{h}_{k-1}^{(j_{k-1})}(x))).$$

To define the power of our pseudorandom generator we need the following notation. Let Q be an arbitrary finite state machine with 2^w states over the alphabet $\{0,1\}^n$. Let D be any distribution over the strings of length $b^k \cdot n$, encoding b^k steps of the FSM. Let $Q(D)$ be a $2^w \times 2^w$ matrix where $[Q(D)]_{ij}$ is the probability that the FSM starting in state i goes to state j after performing b^k steps based on an input drawn from D. Let U_n denote the uniform distribution over $\{0,1\}^n$. We will prove the following lemma.

Lemma 10.3.1. *There exists a constant $c > 0$, given integers n and $w \leq cn$ and parameters b, k with*

$b^k \le 2^{cn}$, for any FSM Q with at most 2^w states, if $\boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1} : \{0, \dots, b-1\} \to \{0,1\}^n \to \{0,1\}^n$ are drawn independently from $\mathcal{H}^b$, where $\mathcal{H}$ is any family of 2-wise independent hash functions, then with probability $\ge 1 - 2^{-cn}$ (over the draw of $\boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1}$),

$$\|Q(G_k(*, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})) - Q((U_n)^{b^k})\| \le 2^{-cn}.$$

Here $G_k(*, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})$ denotes uniform distribution over the set $\{G_k(x, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1}) \mid x \in \{0,1\}^n\}$.

By definition of the matrix norm $\|\cdot\|$ (see section 10.2) this implies that with probability at least $1 - 2^{-cn}$ over the hash functions $\boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1}$, we have that the total variation distance between the distribution of final state using a random string drawn from $(U_n)^{b^k}$ and a random string drawn from $G_k(*, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})$ is at most 2^{-cn}.

Using the above lemma, we will then prove the following theorem.

Theorem 10.3.2. *There exists a constant $c > 0$ such that given any parameters n, b and k satisfying $b^k \le 2^{cn}$, there exists a generator which we call HashPRG $: \{0,1\}^{O(bkn)} \to \{0,1\}^{b^k \cdot n}$ such that HashPRG is an $O(2^{-cn})$ pseudorandom generator for the class of Finite State Machines over alphabet $\{0,1\}^n$ with at most 2^{cn} states. For any seed $\boldsymbol{r}$, the i-th block of bits in HashPRG$(\boldsymbol{r})$ can be computed in time $O(k)$ on a machine with Word Size $\Omega(n)$.*

Proof. Let $p_{i,j}^{h_0,\dots,h_{k-1}}$ be $[Q(G_k(*, h_0, \dots, h_{k-1}))]_{i,j}$. Let $p_{i,j}$ be the probability the FSM starting in state i goes to a state j after b^k steps on an input $\boldsymbol{r}$ where $\boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1} \sim \mathcal{H}^b$ and $\boldsymbol{r} \sim G_k(*, \boldsymbol{h}_0, \dots, \boldsymbol{h}_{k-1})$. Clearly, for any pair of states i, j,

$$p_{i,j} = \frac{1}{|\mathcal{H}^b|^k} \sum_{(h_0,\dots,h_{k-1}) \in (\mathcal{H}^b)^k} p_{i,j}^{h_0,\dots,h_{k-1}}.$$

Let $q_{i,j} = [Q((U_n)^{b^k})]_{i,j}$. Now, for any i,

$$\begin{aligned}\sum_j |p_{i,j} - q_{i,j}| &\le \sum_j \frac{1}{|\mathcal{H}^b|^k} \sum_{(h_0,\dots,h_{k-1}) \in (\mathcal{H}^b)^k} |p_{i,j}^{h_0,\dots,h_{k-1}} - q_{i,j}| \\ &= \frac{1}{|\mathcal{H}^b|^k} \sum_{(h_0,\dots,h_{k-1}) \in (\mathcal{H}^b)^k} (\sum_j |p_{i,j}^{h_0,\dots,h_{k-1}} - q_{i,j}|).\end{aligned}$$

By Lemma 10.3.1, $|\{(h_0, \dots, h_{k-1}) \mid \sum_j |p_{i,j}^{h_0,\dots,h_{k-1}} - q_{i,j}| \le 2^{-cn}\}| \ge |\mathcal{H}^b|^k (1 - 2^{-cn})$ and using the fact that for any $(h_0, \dots, h_{k-1})$, $|p_{i,j}^{h_0,\dots,h_{k-1}} - q_{i,j}| \le 1$, we obtain that

$$\sum_j |p_{i,j} - q_{i,j}| \le 2^{-cn} + 1 \cdot 2^{-cn} \le 2 \cdot 2^{-cn}.$$

As the above holds for any i, we obtain that $\|Q(\text{HashPRG}) - Q((U_n)^{b^k})\| \le 2 \cdot 2^{-cn}$. Where we overload the notation HashPRG to also denote the distribution of string $\boldsymbol{y}$ obtained by first sampling $\boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1}$ and then sampling $\boldsymbol{y} \sim G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})$. The $O(bkn)$ bits of random seed is used to sample bk independent hash functions from the 2-wise independent hash family construction of Dietzfelbinger [Die96, Theorem 3(b)]. The bk independent hash functions are used to construct the hash functions $\boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1}$ and n bits from the random string are used to get an $\boldsymbol{x} \sim \{0,1\}^n$. The output of HashPRG is then $G_k(\boldsymbol{x}, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})$. Clearly, by the definition of G_k, any block of bits in the output string can be computed in $O(k)$ time. □

A special case of the above theorem we use is when $n = O(\log d)$. By choosing $b = d^{\varepsilon}$ and $k = O(1/\varepsilon)$ for a small enough constant ε, we obtain that HashPRG with these parameters fools any FSM with $\mathrm{poly}(d)$ states over an alphabet $\{0,1\}^n$. And that on a machine with word size $\Omega(\log d)$, any block of bits in the output of HashPRG can be computed in time $O(1)$ since ε is a constant.

10.3.2 HashPRG Analysis

We will be reasoning about matrices that represent transition probabilities of Q under different input distributions. We start with some simple facts about norms of matrices. Recall that $\|M\| \coloneqq \sup_{x:\|x\|_\infty=1} \|Mx\|_\infty$. First, for any two matrices A and B we have $\|A+B\| \le \|A\| + \|B\|$ and $\|AB\| \le \|A\|\|B\|$. We will also need the following lemma about the norm of differences of matrix powers:

Lemma 10.3.3. *For integer $b \ge 1$ and square real matrices M and N with $\|M\| \le 1$ and $\|N\| \le 1$, $\|M^b - N^b\| \le b\|M - N\|$.*

Proof. The proof is by induction on b. The statement clearly holds for $b = 1$. For the induction step we have:

$$\begin{aligned}
\|M^b - N^b\| &\le \|M^b - N^{b-1}M\| + \|N^{b-1}M - N^b\| \\
&= \|(M^{b-1} - N^{b-1})M\| + \|N^{b-1}(M - N)\| \\
&\le \|M^{b-1} - N^{b-1}\|\|M\| + \|N\|^{b-1}\|M - N\| \\
&\le (b-1)\|M - N\| + \|M - N\| = b\|M - N\| \ .
\end{aligned}$$

This first uses triangle inequality, the second inequality uses that the norm of a product is bounded by the product of the norms, and the third inequality uses the induction hypothesis as well as the assumption $\|M\| \le 1$, $\|N\| \le 1$. □

For fixed hash functions $h_0, \ldots, h_{k-1}$, let $G_k(*, h_0, \ldots, h_{k-1})$ be the distribution of the bitstring $G_k(\boldsymbol{x}, h_0, \ldots, h_{k-1})$ when $\boldsymbol{x}$ is drawn uniformly at random from $\{0,1\}^n$, denoted $\boldsymbol{x} \sim U_n$. Let $(U_n)^{b^k}$ denote the uniform distribution over length $n \cdot b^k$ bitstrings. The aim is to show, akin to Nisan's generator, for any "small-space computation", with high probability over $\boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1}$, the distribution $G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})$ is indistinguishable from the uniform distribution $(U_n)^{b^k}$.

Lemma 10.3.1 will be derived from this result: For any $\varepsilon > 0$, if all $\boldsymbol{h}_i^{(j)}$ are drawn independently from a 2-wise independent hash family,

$$\mathbf{Pr}\left[\|Q(G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})) - Q((U_n)^{b^k})\| > \varepsilon\right] \le \frac{(b^k - 1)^2 k}{(b-1)^2 \varepsilon^2} \cdot 2^{3w-n} \quad . \tag{10.1}$$

Note that each $\boldsymbol{h}_i$ can also be seen as function from $\{0,1\}^n$ to $\{0,1\}^{bn}$ defined as $\boldsymbol{h}_i(x) = \boldsymbol{h}_i^{(0)}(x) \circ \cdots \circ \boldsymbol{h}_i^{(b-1)}(x)$. We say $\boldsymbol{h}_i : \{0,1\}^n \to \{0,1\}^{bn}$ is drawn from the hash family $\mathscr{H}^b$. Since each $\boldsymbol{h}_i^{(j)}$ is sampled independently from a 2-wise independent hash family, $\mathscr{H}^b$ is also 2-wise independent.

Definition 10.3.4. Let $A \subseteq \{0,1\}^{bn}$, $h : \{0,1\}^n \to \{0,1\}^{bn}$, and $\varepsilon > 0$. We say that h is (ε, A)-*independent* if $|\mathbf{Pr}_{\boldsymbol{x}\sim U_n}[h(\boldsymbol{x}) \in A] - \varrho(A)| \le \varepsilon$, where $\varrho(A) \coloneqq |A|/2^{bn}$ is the density of set A.

The above definition corresponds to Definition 3 in [Nis92] but with some important differences. The differences lead to subtle changes in the whole analysis, but the overall structure of the analysis remain the same.

We now have the following lemma that corresponds to Lemma 1 of [Nis92].

Lemma 10.3.5. *Let $A \subseteq \{0,1\}^{bn}$ and $\varepsilon > 0$. Then $\mathbf{Pr}_{\boldsymbol{h}\sim\mathscr{H}^b}[\boldsymbol{h}$ is not (ε, A)-independent$] < \frac{\varrho(A)}{\varepsilon^2 2^n}$.*

Proof. Consider a matrix M that has a row for each $x \in \{0,1\}^n$ and a column for each $h \in \mathscr{H}^b$. Let $M(x,h) = 1$ if $h(x) \in A$ and 0 otherwise. We define the function f that expresses the probability that a function h maps $\boldsymbol{x} \sim U_n$ to a value in A:

$$f(h) = \mathbf{E}_{\boldsymbol{x}\sim U_n}[M(\boldsymbol{x}, h)] = \mathbf{Pr}_{\boldsymbol{x}\sim U_n}[h(\boldsymbol{x}) \in A].$$

For $\boldsymbol{h} \sim \mathscr{H}^b$ we have

$$\mathbf{E}_{\boldsymbol{h}\sim\mathscr{H}^b} f(\boldsymbol{h}) = \mathbf{E}_{\boldsymbol{h}\sim\mathscr{H}^b, \boldsymbol{x}\sim U_n}[M(\boldsymbol{x}, \boldsymbol{h})] = \mathbf{E}_{\boldsymbol{x}\sim U_n} \mathbf{Pr}_{\boldsymbol{h}\sim\mathscr{H}^b}[\boldsymbol{h}(\boldsymbol{x}) \in A] = \varrho(A),$$

where the last equality follows from $\mathbf{Pr}_{\boldsymbol{h}\sim\mathscr{H}^b}[\boldsymbol{h}(\boldsymbol{x}) \in A] = \varrho(A)$, since $\mathscr{H}^b$ is 2-wise independent. We next bound the variance of $f(\boldsymbol{h})$ to show that $f(\boldsymbol{h})$ is close to $\varrho(A)$ with high probability:

$$\begin{aligned}\mathbf{Var}_{\boldsymbol{h}\sim\mathscr{H}^b} f(\boldsymbol{h}) &= \mathbf{E}_{\boldsymbol{h}\sim\mathscr{H}^b}(f(\boldsymbol{h}) - \varrho(A))^2 \\ &= \mathbf{E}_{\boldsymbol{h}\sim\mathscr{H}^b}(\mathbf{E}_{\boldsymbol{x}\sim U_n}[M(\boldsymbol{x}, \boldsymbol{h}) - \varrho(A)])^2.\end{aligned}$$

If $\boldsymbol{x}_1 \sim U_n$ and $\boldsymbol{x}_2 \sim U_n$ are independently drawn we can expand

$$\begin{aligned}&(\mathbf{E}_{\boldsymbol{x}\sim U_n}[M(\boldsymbol{x}, h) - \varrho(A)])^2 \\ &= \mathbf{E}_{\boldsymbol{x}_1\sim U_n, \boldsymbol{x}_2\sim U_n}(M(\boldsymbol{x}_1, h) - \varrho(A))(M(\boldsymbol{x}_2, h) - \varrho(A)).\end{aligned}$$

Using the fact that for every x, $\mathbf{E}_{\boldsymbol{h}\sim\mathcal{H}^b}[M(x,\boldsymbol{h})] = \varrho(A)$ we obtain

$$\mathbf{Var}_{\boldsymbol{h}\sim\mathcal{H}^b} f(\boldsymbol{h}) = \mathbf{E}_{\boldsymbol{x}_1\sim U_n,\boldsymbol{x}_2\sim U_n}\mathbf{E}_{\boldsymbol{h}\sim\mathcal{H}^b}[M(\boldsymbol{x}_1,\boldsymbol{h})M(\boldsymbol{x}_2,\boldsymbol{h}) - \varrho(A)^2].$$

With probability $1-2^{-n}$ we have $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$. Since $\boldsymbol{h}$ is drawn from a 2-wise independent hash family, $\boldsymbol{h}(\boldsymbol{x}_1)$ and $\boldsymbol{h}(\boldsymbol{x}_2)$ are independent in this case, so $\mathbf{E}_{\boldsymbol{h}\sim\mathcal{H}^b}[M(\boldsymbol{x}_1,\boldsymbol{h})M(\boldsymbol{x}_2,\boldsymbol{h}) \mid \boldsymbol{x}_1 \neq \boldsymbol{x}_2] = \varrho(A)^2$. With probability $1/2^n$, we have $\boldsymbol{x}_1 = \boldsymbol{x}_2$ and $\mathbf{E}_{\boldsymbol{h}\sim\mathcal{H}^b}[M(\boldsymbol{x}_1,h)M(\boldsymbol{x}_2,h) \mid \boldsymbol{x}_1 = \boldsymbol{x}_2] = \mathbf{E}_{\boldsymbol{h}\sim\mathcal{H}^b}[M(\boldsymbol{x}_1,\boldsymbol{h})] = \varrho(A)$. Hence,

$$\mathbf{Var}_{\boldsymbol{h}\sim\mathcal{H}^b} f(\boldsymbol{h}) = (1-2^{-n})\varrho(A)^2 + 2^{-n}\varrho(A) - \varrho(A)^2 < \frac{\varrho(A)}{2^n}.$$

By Chebyshev's inequality,

$$\mathbf{Pr}_{\boldsymbol{h}\sim\mathcal{H}^b}[|f(\boldsymbol{h}) - \varrho(A)| \geq \varepsilon] \leq \frac{\varrho(A)}{2^n\varepsilon^2}. \qquad \square$$

We now have the following definition that corresponds to Definition 4 of [Nis92].

Definition 10.3.6. Let Q be an FSM on alphabet $\{0,1\}^n$ and pick $\varepsilon > 0$. For hash functions $h_0,\ldots,h_{k-1} : \{0,1\}^n \to \{0,1\}^{bn}$, we say $(h_0,\ldots,h_{k-1})$ is *(ε, Q)-good* if $\|Q(G_k(*,h_0,\ldots,h_{k-1})) - Q((U_n)^{b^k})\| \leq \varepsilon$.

For small ε the definition essentially says that the FSM Q cannot distinguish between the distributions $G_k(*,h_0,\ldots,h_{k-1})$ and $(U_n)^{b^k}$. The following lemma corresponds to Lemma 2 of [Nis92].

Lemma 10.3.7. *Let Q be an FSM of size 2^w over alphabet $\{0,1\}^n$ and k a nonnegative integer. Then for every $\varepsilon > 0$,*

$$\begin{aligned}&\mathbf{Pr}_{\boldsymbol{h}_0,\ldots,\boldsymbol{h}_{k-1}\sim\mathcal{H}^b}[(\boldsymbol{h}_0,\ldots,\boldsymbol{h}_{k-1}) \textit{ is not } ((b^k-1)\varepsilon, Q)\textit{-good}]\\ &\leq \frac{2^{3w}k}{(b-1)^2\varepsilon^2 2^n}.\end{aligned}$$

Proof. The proof is a careful translation of the proof of Lemma 2 of [Nis92]. The proof is by induction on k. For $k = 0$, the statement is immediate since $G_0(x) \sim U_n$ which means that there is zero difference between the distributions. For the induction step assume that the statement holds for $k-1$. For every choice of $h_0,\ldots,h_{k-2} : \{0,1\}^n \to \{0,1\}^{bn}$ and every two states i, j of Q we define the set of seeds $y^{(0)},\ldots,y^{(b-1)}$ that when used one by one with G_{k-1} produces a string that takes Q from state i to state j:

$$\begin{aligned}B_{i,j}^{h_0,\ldots,h_{k-2}} = \{&(y^{(0)},\ldots,y^{(b-1)}) \in \{0,1\}^{bn} \mid G_{k-1}(y^{(0)},h_0,\ldots,h_{k-2}) \circ \cdots \circ G_{k-1}(y^{(b-1)},h_0,\ldots,h_{k-2})\\ &\text{takes } Q \text{ from } i \text{ to } j\}.\end{aligned}$$

Now sample $\boldsymbol{h}_0,\ldots,\boldsymbol{h}_{k-1} \sim \mathcal{H}^b$ independently and consider the following events:

1. $(\boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2})$ is $((b^{k-1}-1)\varepsilon, Q)$-good (see Definition 10.3.6), and
2. $\boldsymbol{h}_{k-1}$ is $(\varepsilon(b-1)/2^w, B_{i,j}^{\boldsymbol{h}_0,\ldots,\boldsymbol{h}_{k-2}})$-independent for all states i, j (see Definition 10.3.4).

We will see that these events happen simultaneously with probability at least $1 - \frac{2^{3w}k}{(b-1)^2\varepsilon^2 2^n}$. Furthermore, we will show that when this happens, $(\boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})$ is $((b^k-1)\varepsilon, Q)$-good, completing the induction step.

By the induction hypothesis, the probability of event 1 *not* happening is at most $\frac{2^{3w}(k-1)}{(b-1)^2\varepsilon^2 2^n}$. From Lemma 10.3.5, for every choice of $h_0, \ldots, h_{k-2}$ and states i, j of Q, the probability that $\boldsymbol{h}_{k-1}$ is *not* $(\varepsilon(b-1)/2^w, B_{i,j}^{h_0,\ldots,h_{k-2}})$-independent is at most

$$\frac{\varrho(B_{i,j}^{h_0,\ldots,h_{k-2}})2^{2w}}{\varepsilon^2(b-1)^2 2^n} \quad .$$

Using a union bound, the probability there exists a pair of states i and j such that $\boldsymbol{h}_{k-1}$ is not $(\varepsilon(b-1)/2^w, B_{i,j}^{h_0,\ldots,h_{k-2}})$-independent is at most

$$\sum_{i,j} \frac{\varrho(B_{i,j}^{h_0,\ldots,h_{k-2}})2^{2w}}{\varepsilon^2(b-1)^2 2^n} = \frac{2^{2w}}{\varepsilon^2(b-1)^2 2^n} \sum_{i,j} \varrho(B_{i,j}^{h_0,\ldots,h_{k-2}}) \leq \frac{2^{3w}}{\varepsilon^2(b-1)^2 2^n} \quad .$$

The last inequality follows from the fact that for every fixed i, the sets $B_{i,j}^{h_0,\ldots,h_{k-2}}$, where j ranges over the states of Q, partition $\{0,1\}^{bn}$ and thus their ϱ-values sum up to 1. A union bound shows us that the probability of either event 1 or 2 not holding is at most $2^{3w}k/(b-1)^2\varepsilon^2 2^n$, as claimed.

Let $(G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}))^b$ denote the distribution over bitstrings of length $n \cdot b^k$ obtained by concatenating b *independent* samples drawn from $G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2})$. Conditioning on events 1 and 2 we now bound $\|Q(G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1}) - Q((U_n)^{b^k})\|$. Using the triangle inequality,

$$\begin{aligned}\|Q(G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})) - Q((U_n)^{b^k})\| \leq{} & \|Q(G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})) - Q((G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}))^b)\| \\ & + \|Q((G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}))^b) - Q((U_n)^{b^k})\|.\end{aligned}$$

Below we will bound the first term by $(b-1)\varepsilon$ and second term by $(b^k-b)\varepsilon$, proving that the hash functions are $((b^k-1)\varepsilon, Q)$-good as claimed.

Consider the matrix $Q(G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1}))$. By definition, entry (i, j) of the matrix is equal to $\mathbf{Pr}_{\boldsymbol{x}\sim U_n}[\boldsymbol{h}_{k-1}(\boldsymbol{x}) \in B_{i,j}^{\boldsymbol{h}_0,\ldots,\boldsymbol{h}_{k-2}}]$. Now consider $Q((G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}))^b)$ where entry (i, j) is

$$\mathbf{Pr}_{\boldsymbol{y}^{(0)},\ldots,\boldsymbol{y}^{(b-1)}\sim U_n}[G_{k-1}(\boldsymbol{y}^{(0)}, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}) \circ \cdots \circ G_{k-1}(\boldsymbol{y}^{(b-1)}, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}) \text{ takes } Q \text{ from } i \text{ to } j] \quad .$$

By definition, the above quantity is exactly $\varrho(B_{i,j}^{\boldsymbol{h}_0,\ldots,\boldsymbol{h}_{k-2}})$. Since event 2 holds, we have that the absolute value of every entry of the matrix $Q(G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})) - Q((G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}))^b)$ is at most $\varepsilon(b-1)/2^w$. Since each row has at most 2^w entries we obtain

$$\|Q(G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})) - Q((G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}))^b)\| \leq (b-1)\varepsilon \ .$$

To bound the second term we define the transition matrices $M = Q(G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}))$ and $N = Q((U_n)^{b^{k-1}})$ that describe transition probabilities using G_{k-1} and uniform inputs, respectively. Since the b parts of the input are independent we can express the matrices in the second term as powers of M and N:

$$Q((G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1}))^b) = M^b \quad \text{and} \quad Q((U_n)^{b^k}) = N^b \ .$$

We can now invoke Lemma 10.3.3. Since event 1 holds we have $\|M - N\| \leq (b^{k-1}-1)\varepsilon$, and thus

$$\|Q((G_{k-1}(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-2}))^b) - Q((U_n)^{b^k})\| \leq b(b^{k-1}-1)\varepsilon = (b^k - b)\varepsilon.$$

Conditioning on events 1 and 2 we thus have $\|Q(G_k(*, \boldsymbol{h}_0, \ldots, \boldsymbol{h}_{k-1})) - Q((U_n)^{b^k})\| \leq (b^k - 1)\varepsilon$. $\square$

The proof of Lemma 10.3.1 now follows by choosing a small constant c and setting $w = cn$ and $\varepsilon = 2^{-cn}/(b^k - 1)$ in the above lemma. From Proposition 1 of [Nis92], it follows that any space(cn) algorithm that uses nb^k uniform random bits, for $b^k \leq 2^{cn}$, can use the bits from the pseudorandom generator and with probability at least $1 - 2^{-cn}$, the total variation distance of the final state of the algorithm using pseudorandom bits to the final state of the algorithm using uniform random bits is at most 2^{-cn}.

10.4 Moment Estimation for $p > 2$

Using min-stability of exponential random variables, Andoni [And17] gave Algorithm 10.1 to estimate F_p moments in the stream. We state their result below and describe their algorithm.

Consider the vector x obtained at the end of the stream applying all the updates sequentially to the starting vector 0. As Algorithm 10.1 is linear, we have that the final state of the algorithm depends only on the final vector and not on the order of the updates. To estimate the ℓ_p norm of a d-dimensional vector x, the algorithm first samples d independent standard exponential random variables $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_d$ and creates a random vector $\boldsymbol{z} \in \mathbb{R}^d$ such that $\boldsymbol{z}_i = \boldsymbol{E}_i^{-1/p} x_i$. Andoni shows that $\|\boldsymbol{z}\|_\infty = \Theta(\|x\|_p)$ and therefore estimating the value of the coordinate in $\boldsymbol{z}$ with maximum absolute value gives a constant factor approximation for $\|x\|_p$. Andoni further shows that with high probability there are only at most $O(\log d)^p$ coordinates in $\boldsymbol{z}$ with absolute value $\geq \|x\|_p/(c \log d)$ for a constant c hence showing that the vector $\boldsymbol{z}$ has only a few coordinates with large values. The final property of $\boldsymbol{z}$ that Andoni uses is that with high probability $\|\boldsymbol{z}\|_2^2 = O(d^{1-2/p}\|x\|_p^2)$.

Conditioned on the above properties of $\boldsymbol{z}$, Andoni argues that if $\boldsymbol{S}$ is a CountSketch matrix with

$O(d^{1-2/p}\log d)$ rows formed using $O(\log d)$-wise independent hash functions, then with high probability $\|\boldsymbol{S}\boldsymbol{z}\|_\infty = \Theta(\|\boldsymbol{z}\|_\infty) = \Theta(\|x\|_p)$. To implement the streaming algorithm in sublinear space we cannot store the exponential random variables and the vector $\boldsymbol{z}$ in the stream. So Andoni derandomizes his algorithm by using a pseudorandom generator of Nisan and Zuckerman [NZ96] and shows that exponential random variables $\boldsymbol{E}_i$ generated using the pseudorandom bits are sufficient to ensure that the algorithm produces a constant factor approximation to $\|x\|_p$. Also note that the algorithm does not need to explicitly store the d dimensional vector $\boldsymbol{z}$; it is enough just to update the $O(d^{1-2/p}\log d)$ dimensional vector $\boldsymbol{f}$ in the stream.

Although Andoni's algorithm is space optimal for linear sketches up to constant factors, the update time using the Nisan-Zuckerman pseudorandom generator is $\text{poly}(d)$. We now show that Andoni's algorithm can be derandomized using our HashPRG to obtain an algorithm that is both space optimal for linear sketches and has an update time of $O(1)$ in the Word RAM model with a word size $\Omega(\log d)$.

Algorithm 10.1: Andoni's Algorithm with Independent Exponentials [And17]

```
Input: $p > 2$, $d \in \mathbb{N}$, a stream of updates $(i_1, v_1), \dots, (i_m, v_m) \in [d] \times \{-M, \dots, M\}$ for
       $m, M = \text{poly}(d)$
Output: An approximation to $\|x\|_p$ where $x \in \mathbb{R}^d$ is defined by the stream of updates
1  $\boldsymbol{E}_1, \dots, \boldsymbol{E}_d \leftarrow$ Independent standard exponential random variables;
2  $d' \leftarrow O(d^{1-2/p}\log d)$;
3  $\boldsymbol{z} \leftarrow 0_d, \boldsymbol{f} \leftarrow 0_s$;
4  $\boldsymbol{h} \leftarrow O(\log d)$-wise independent hash function from $[d]$ to $[d']$;
5  $\boldsymbol{\sigma} \leftarrow O(\log d)$-wise independent hash function from $[d]$ to $\{-1, +1\}$;
6  for $j = 1, \dots, m$ do
7      $\boldsymbol{z}_{i_j} \leftarrow \boldsymbol{z}_{i_j} + \boldsymbol{E}_i^{-1/p} v_j$;
8      $\boldsymbol{f}_{\boldsymbol{h}(i_j)} \leftarrow \boldsymbol{f}_{\boldsymbol{h}(i_j)} + \boldsymbol{\sigma}(i)\boldsymbol{E}_i^{-1/p} v_j$;
9  end
10 return $\|\boldsymbol{f}\|_\infty$;
```

10.4.1 Discretizing the Exponentials

We first show that we can replace exponential random variables in Andoni's algorithm with a simple discrete random variable and obtain all the guarantees we stated above that $\boldsymbol{z}$ satisfies even with this discrete random variable. For now, assume that $p = 1$. We will later generalize the guarantees for all p.

Suppose $x \in \mathbb{R}^d$ with all the coordinates of x being integers with absolute values $\le \text{poly}_1(d)$. We can assume x has no nonzero coordinates. Consider the discrete random variable $\boldsymbol{E}$ that takes values in the set $\{1, 2, \ldots, 2^M\}$ for some $M = O(\log d)$ satisfying $2^M \geq d\,\mathrm{poly}_1(d)$. Let

$$\mathbf{Pr}[\boldsymbol{E} = 2^j] = \begin{cases} 1/2^{j+1} & 0 \leq j < M \\ 1/2^M & j = M. \end{cases}$$

We call the random variable $\boldsymbol{E}$ a *discrete exponential*[5]. Note that we can sample the random variable $\boldsymbol{E}$ quite easily from a uniform random bitstring of length M just based on the position of the first appearance of 1 in the random bitstring. We mainly use the following properties of $\boldsymbol{E}$: for any $t > 0$, $\mathbf{Pr}[\boldsymbol{E} \geq t] \leq 1/t$. The statement clearly holds for $t \leq 1$. For $t \geq 1$, let 2^j be such that $t \leq 2^j < 2t$. Now,

$$\begin{aligned}\mathbf{Pr}[\boldsymbol{E} \geq t] &= \mathbf{Pr}[\boldsymbol{E} = 2^j] + \mathbf{Pr}[\boldsymbol{E} = 2^{j+1}] + \cdots \\ &= \frac{1}{2^{j+1}} + \frac{1}{2^{j+2}} + \cdots = \frac{1}{2^j} \leq \frac{1}{t}.\end{aligned}$$

Similarly, for any $t \leq 2^M$, we have $\mathbf{Pr}[\boldsymbol{E} \geq t] \geq \min(1, 1/(2t))$. We now prove the following lemma.

Lemma 10.4.1. *Let $x \in \mathbb{R}^d$ be an arbitrary vector with integer entries of absolute value at most* $\mathrm{poly}_1(d)$. *Let $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_d$ be independent* discrete exponential *random variables. Then,*

1. *With probability* $\geq 95/100$,

$$\frac{\|x\|_1}{16} \leq \max_i |x_i|\boldsymbol{E}_i \leq 50\|x\|_1.$$

2. *For any T,*

$$\mathbf{E}[|\{i \mid |x_i|\boldsymbol{E}_i \geq \|x\|_1/T\}|] \leq T.$$

Proof. Without loss of generality, we can assume that all the coordinates of x are nonzero. Using the distribution of the random variable $\boldsymbol{E}_i$, we obtain that

$$\mathbf{Pr}\left[\boldsymbol{E}_i \geq 50\frac{\|x\|_1}{|x_i|}\right] \leq \frac{|x_i|}{50\|x\|_1}.$$

Thus, by a union bound, with probability $\geq 1 - 1/50$, for all i, $\boldsymbol{E}_i \leq 50\|x\|_1/|x_i|$. Hence, with probability $\geq 49/50$, $\max_i |x_i|\boldsymbol{E}_i \leq 50\|x\|_1$. Similarly, for all i,

$$\mathbf{Pr}\left[\boldsymbol{E}_i \geq \frac{\|x\|_1}{16|x_i|}\right] \geq \min(8|x_i|/\|x\|_1, 1).$$

If there exists an index i such that $|x_i| \geq \|x\|_1/8$, then we already have that with probability 1,

[5]While $\boldsymbol{E}$ models the inverse of a continuous exponential random variable, we use the term "discrete exponential" for brevity.

$\max_i |x_i|\boldsymbol{E}_i \ge \|x\|_1/8$. Now assume that for all i, $|x_i| \le \|x\|_1/8$. Using the independence of $\boldsymbol{E}_i$'s, we obtain that with probability $\ge 99/100$, there exists an index i such that $\boldsymbol{E}_i \ge \|x\|_1/(16|x_i|)$ and therefore with probability $\ge 99/100$, $\max_i |x_i|\boldsymbol{E}_i \ge \|x\|_1/16$. Hence, overall with probability $\ge 95/100$, we have

$$\frac{\|x\|_1}{16} \le \max_i |x_i|\boldsymbol{E}_i \le 50\|x\|_1.$$

Let $T > 0$ be arbitrary. For a fixed i, we have $\mathbf{Pr}[\boldsymbol{E}_i \ge \|x\|_1/(|x_i|T)] \le |x_i|T/\|x\|_1$. Thus,

$$\begin{aligned}&\mathbf{E}[|\{i \mid |x_i|\boldsymbol{E}_i \ge \|x\|_1/T\}|]\\ &= \sum_i \mathbf{Pr}[\boldsymbol{E}_i\\ &\ge \|x\|_1/(|x_i|T)] \le T\sum_i |x_i|/\|x\|_1 = T.\end{aligned}$$ □

The following lemma extends the above properties to all $p > 2$ along with an additional property that Andoni uses in his proof.

Lemma 10.4.2. *Let $p > 2$ be arbitrary and let $x \in \mathbb{R}^d$ be an arbitrary vector of integer entries with absolute value at most $(\text{poly}_1(d))^{1/p}$. Let $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_d$ be independent discrete exponential random variables. Then,*

1. *With probability $\ge 95/100$,*

$$\frac{\|x\|_p}{16^{1/p}} \le \max_i |x_i|\boldsymbol{E}_i^{1/p} \le 50^{1/p}\|x\|_p.$$

2. *For any $T > 0$, with probability $\ge 95/100$,*

$$|\{i \mid |x_i|\boldsymbol{E}_i^{1/p} \ge \|x\|_p/T^{1/p}\}| \le 20T.$$

3.

$$\mathbf{E}[\sum_{i=1}^d (\boldsymbol{E}_i^{1/p}x_i)^2] = O_p(d^{1-2/p}\|x\|_p^2).$$

Proof. The first two properties follow from applying the previous lemma to the vector $x^{(p)} \in \mathbb{R}^d$ defined as $x_i^{(p)} = |x_i|^p$. To prove the last property, we have

$$\mathbf{E}[\sum_{i=1}^d (\boldsymbol{E}_i^{1/p}x_i)^2] = \sum_{i=1}^d x_i^2\,\mathbf{E}[\boldsymbol{E}_i^{2/p}].$$

Now $\mathbf{E}[\boldsymbol{E}_i^{2/p}] = \sum_{j=0}^{M-1} 2^{2j/p}/2^{j+1} + 2^{2M/p}/2^M \le (1/2)\sum_{j=0}^{\infty} 2^{j(2/p-1)} + 2^{M(2/p-1)} \le 1/(2-2^{2/p}) + 1.$

Hence,

$$\begin{aligned}\mathbf{E}[\sum_{i=1}^{d}(\boldsymbol{E}_i^{1/p}x_i)^2] &\le \|x\|_2^2\left(\frac{1}{2-2^{2/p}}+1\right)\\ &= O_p(d^{1-2/p}\|x\|_p^2)\end{aligned}$$

where the constant we are hiding blows up as $p \to 2$. □

10.4.2 F_p Estimation With HashPRG

In the previous section, we discussed some properties satisfied by the random vector $\boldsymbol{z}$ given by $\boldsymbol{z}_i \coloneqq \boldsymbol{E}_i^{1/p}x_i$ for any arbitrary vector x with integer entries of absolute value at most $(\mathrm{poly}_1(d))^{1/p}$. Now we show that the properties are still satisfied even when the random variables $\boldsymbol{E}_i$ are generated using HashPRG thereby showing that Algorithm 10.2 outputs a constant factor approximation to $\|x\|_p$ with probability $\ge 7/10$. The success probability can be increased to $1-\delta$ by running $O(\log 1/\delta)$ independent copies of the algorithm and reporting the median.

Theorem 10.4.3. *Let $p > 2$ be a parameter and for $m = O(\mathrm{poly}(d))$ let the vector $x \coloneqq 0 \in \mathbb{R}^d$ receive a stream of m updates $(i_1, v_1), (i_2, v_2), \ldots, (i_m, v_m)$ with $|v_j| \le \mathrm{poly}(d)$ for all j. On receiving an update (i_j, v_j), the vector x is modified as $x_{i_j} \leftarrow x_{i_j} + v_j$. The Algorithm 10.2 uses $O_p(d^{1-2/p}\log(d))$ words of space and at the end of the stream outputs a constant factor approximation to $\|x\|_p$ with probability $\ge 7/10$. Further each update to x is processed by the algorithm in $O(1)$ time in the Word RAM model on a machine with a word size $\Omega(\log d)$.*

Proof. First we condition on the event that the hash families from which $\boldsymbol{h}, \boldsymbol{\sigma}$ are drawn are $O(\log d)$-wise independent. From Theorem 10.2.2, the event holds with probability $\ge 99/100$. From Theorem 10.3.2, HashPRG with parameters $n = O(\log d)$, $b = d^{\varepsilon}$ and $k = O(1/\varepsilon)$ fools a Finite State Machine with $\mathrm{poly}(d)$ states. Further the seed for HashPRG can be stored using $O((1/\varepsilon)d^{\varepsilon}\log d) = o(d^{1-2/p})$ bits if $\varepsilon < 1 - 2/p$. Theorem 10.2.2 shows that $\boldsymbol{h}$ and $\boldsymbol{\sigma}$ can be stored using $O(d^{\varepsilon}) = o(d^{1-2/p})$ bits of space and for any $i \in [d]$, $\boldsymbol{h}(i)$ and $\boldsymbol{\sigma}(i)$ can be evaluated in $O_{1/\varepsilon}(1)$ time. Therefore, each update in the stream is processed in $O_{1/\varepsilon}(1)$ time.

Let x be the vector at the end of the stream and for $i \in [d]$, $\boldsymbol{E}_i$ be the discrete exponential random variable computed by the algorithm for coordinate i. Let $T \coloneqq (O(\log d))^p$. Define a Finite State Machine Q_x with a start state and other states being defined by the tuple (i, j, t, v) with $1 \le i \le d$, $0 \le j \le d$, $1 \le v \le \mathrm{poly}(d)$ and $t \in \{\mathrm{less}, \mathrm{correct}, \mathrm{more}\}$. The machine Q_x clearly has $\mathrm{poly}(d)$ states. The FSM being in a state (i, j, less, v) denotes that it has processed the coordinates $x_1, \ldots, x_i$

until now and found that

$$|\{i' \leq i \mid |x_{i'}|\boldsymbol{E}_{i'}^{1/p} \geq \|a\|_p f^{1/p}\}| = j,$$
$$\max_{i' \leq i} |x_{i'}|\boldsymbol{E}_{i'}^{1/p} < \frac{\|x\|_p}{16^{1/p}}, \text{ and}$$
$$\sum_{i' \leq i} \text{round}(\boldsymbol{E}_{i'}^{1/p})^2 \cdot (x_{i'})^2 = v.$$

Here round(x) denotes x rounded to the nearest integer. As $\boldsymbol{E}_{i'}^{1/p} \geq 1$ if $x_{i'} \neq 0$, we have

$$(\boldsymbol{E}_{i'}^{1/p} x_{i'})^2 \leq \text{round}(\boldsymbol{E}_{i'}^{1/p})^2 \cdot (x_{i'})^2 \leq 4(\boldsymbol{E}_{i'}^{1/p} x_{i'})^2.$$

Note that using the bound on the absolute values of entries in x, the value v can be at most poly(d). Similarly, the state is in $(i, j, \text{correct}, v)$ if a condition similar to above holds, but instead we have

$$\frac{\|x\|_p}{16^{1/p}} \leq \max_{i' \leq i} |x_{i'}|\boldsymbol{E}_{i'}^{1/p} \leq 50^{1/p}\|x\|_p,$$

and in (i, j, more, v) if $\max_{i' \leq i} |x_{i'}|\boldsymbol{E}_{i'}^{1/p} > 50^{1/p}\|x\|_p$. Here the value of $\boldsymbol{E}_i$ is assigned based on the bitstring corresponding to an edge in the FSM. It is clear that we can construct such a Finite State Machine. Let $\mathscr{E}$ denote the event that $\max_i |x_i|\boldsymbol{E}_i^{1/p} \in [\|x\|_p/16^{1/p}, 50^{1/p}\|x\|_p]$, $|\{i \mid |x_i|\boldsymbol{E}_i^{1/p} \geq \|x\|_p/T^{1/p}\}| \leq 20T$ and $\sum_{i \leq i} \text{round}((\boldsymbol{E}_i^{1/p})x_i)^2 = O_p(d^{1-2/p}\|x\|_p^2)$. By Lemma 10.4.2, the final state distribution of the FSM using uniform random edge at every state satisfies the event $\mathscr{E}$ with probability $\geq 85/100$. By Theorem 10.3.2, if the random variables $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_d$ are generated using the random string sampled from HashPRG, then with probability $\geq 1 - 1/\text{poly}(d)$, the final state of FSM Q_x satisfies the event $\mathscr{E}$ with probability $\geq 8/10$.

Thus, with probability $\geq 8/10$, the implicit vector $\boldsymbol{z} \in \mathbb{R}^d$ in the algorithm defined as $\boldsymbol{z}_i \coloneqq \boldsymbol{E}_i^{1/p} x_i$ satisfies all the properties Andoni requires of the vector obtained by multiplying coordinates of x with scaled exponential random variables. Hence, with probability $\geq 75/100$, the maximum absolute value of the coordinate in $\boldsymbol{f}$ obtained by sketching $\boldsymbol{z}$ with a CountSketch matrix is a constant factor approximation to $\|x\|_p$.

Thus, overall the algorithm outputs a constant factor approximation to $\|x\|_p$ with probability $\geq 7/10$. □

10.4.3 Comparison with Andoni's use of Nisan-Zuckerman PRG

Andoni argues that his algorithm can be run in $O(d^{1-2/p} \log d)$ words of space using the Nisan-Zuckerman pseudorandom generator, which shows that an S space algorithm using poly(S) random bits can be run with just $O(S)$ random bits. Nisan-Zuckerman's algorithm takes an $O(S)$ length uni-

Algorithm 10.2: F_p moment estimation using HashPRG

```
Input: $p > 2, d \in \mathbb{N}$, a stream of updates $(i_1, v_1), \ldots, (i_m, v_m) \in [d] \times \{-M, \ldots, M\}$ for
       $m, M = \mathrm{poly}(d)$
Output: An approximation to $\|x\|_p$ where $x \in \mathbb{R}^d$ is defined by the stream of updates
1  $\varepsilon \leftarrow$ A constant smaller than $1 - 2/p$;
2  $\boldsymbol{s} \leftarrow$ Pseudorandom string constructed using HashPRG with parameters
     $n = O(\log d), b = d^{\varepsilon}, k = O(1/\varepsilon)$;
   // The string $\boldsymbol{s}$ is only implicitly stored using the corresponding hash
      functions and the random seed to the generator
3  $d' \leftarrow O(d^{1-2/p} \log d)$;
4  $\boldsymbol{h} \leftarrow O(\log d)$-wise independent hash function from $[d] \to [d']$;
5  $\boldsymbol{\sigma} \leftarrow O(\log d)$-wise independent hash function from $[d] \to \{+1, -1\}$;
   // Both $\boldsymbol{h}$ and $\boldsymbol{\sigma}$ are drawn from hash family in Theorem 10.2.2 so that
      they can be stored using $O_{1/\varepsilon}(d^{\varepsilon})$ bits and evaluated in $O_{1/\varepsilon}(1)$ time on
      any input
6  $\boldsymbol{f} \leftarrow 0_m$;
   // Stream Processing Begins
7  for $j = 1, \ldots, m$ do
8      $\boldsymbol{E}_{i_j} \leftarrow \mathrm{DiscreteExponential}(i_j\text{-th chunk of } \boldsymbol{s})$;
9      $\boldsymbol{f}_{\boldsymbol{h}(i_j)} \leftarrow \boldsymbol{f}_{\boldsymbol{h}(i_j)} + \boldsymbol{\sigma}(i_j) \cdot \mathrm{round}(\boldsymbol{E}_{i_j}^{1/p}) \cdot v_j$;
10 end
11 return $\|\boldsymbol{f}\|_\infty$;
```

formly random string and stretches it by a factor of $O(S^\gamma)$ $(0 < \gamma < 1)$ by computing $O(S^\gamma)$ blocks of $O(S)$ bits each. Each of the $O(S^\gamma)$ blocks of pseudorandom bits takes time $\text{poly}(S)$ time to compute which in our case is $\text{poly}(d^{1-2/p} \log d)$ and hence prohibitive.

10.4.4 ℓ_p sampling

As another application of HashPRG, we give a simple ℓ_p sampling algorithm for $p > 2$. Assume the same turnstile stream setting. At the end of the stream, ℓ_p sampling asks to output a coordinate i of the underlying vector x such that probability of sampling i is proportional to $|x_i|^p/\|x\|_p^p$. The problem has been widely studied (see [CJ19, JW18] and references therein). The perfect ℓ_p sampling algorithm of [JW18] for $p \in (0, 2)$ uses the following property of exponential random variables: if $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_d$ are independent standard exponential random variables and $i^* = \arg\max_i x_i/\boldsymbol{E}_i^{1/p}$, then $\mathbf{Pr}[i^* = i] = |x_i|^p/\|x\|_p^p$. This distribution exactly corresponds to what ℓ_p sampling asks. To implement the algorithm in a turnstile stream using a small amount of space, they first scale the coordinates with exponentials and then sketch the scaled vector using a data structure called *count-max* and show that the count-max data structure allows to recover the max coordinate in the vector obtained by scaling x with exponential random variables. Finally, they derandomize their construction using a half-space fooling pseudorandom generator.

We show that using HashPRG, for $p > 2$, we obtain ℓ_p samplers that have a very fast update time. For simplicity, we discuss an algorithm that samples from the following distribution:

$$\mathbf{Pr}[i \text{ is sampled}] \geq \frac{1}{1+\varepsilon}\frac{|x_i|^p}{\|x\|_p^p} \pm 1/\text{poly}(d).$$

In the definition approximate perfect samplers, it is required that $\mathbf{Pr}[i \text{ is sampled}]$ is $(1\pm\varepsilon)|x_i|^p/\|x\|_p^p$ up to an additive $1/\text{poly}(d)$ error. We discuss the simpler version as ℓ_p samplers are not our main focus.

For this, we work with a finer approximation of exponential random variables than what we used for approximating F_p moments. Assume that given a block of $O(\log d)$ uniform random bits, there is a fast way to convert the random bits into a fine enough discretization of the exponential random variables such that all the following property of the exponential random variables hold for this discretization:

1. The probability that $i^* = i$ and $\boldsymbol{E}_{i^*}^{-1/p}|x_{i^*}| \geq (1+\varepsilon)^{-1/p} \max_{i' \neq i} \boldsymbol{E}_i^{-1/p}|x_i|]$ is at least $|x_i|^p/(1+\varepsilon)\|x\|_p^p$, and
2. with probability at least $1 - 1/\text{poly}(d)$,

$$\sum_{i=1}^{d}(\boldsymbol{E}_i^{-1/p}|x_i|)^2 \leq d^{1-2/p}(\boldsymbol{E}_{i^*}^{-1/p}|x_{i^*}|)^2 \,\text{polylog}(d).$$

It can be easily shown that both of these properties hold for *continuous* exponential random variables and hence they hold for a suitable discretization of the continuous exponential random variables. The above properties crucially depend on $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_d$ being independent, but we can derandomize them by using HashPRG. Fix a vector x and design the following FSM for x: the FSM goes through each coordinate of x sequentially. The FSM tracks $\max_i \boldsymbol{E}_i^{-1/p}|x_i|$, the coordinate attaining the max, $\sum_i \boldsymbol{E}_i^{-2/p} x_i^2$. Clearly, all these statistics can be tracked using an FSM with $\operatorname{poly}(d)$ states similar to how we derandomized the properties of exponential random variables for approximating F_p moments. Using $n = O(\log d)$ and $b = d^c$ ($c < 1 - 2/p$) for HashPRG, thus we obtain that if the random variables $\boldsymbol{E}_i$ are generated using the pseudorandom string sampled from HashPRG, then for all i,

$$\begin{aligned}
&\mathbf{Pr}_{(\boldsymbol{E}_1,\ldots,\boldsymbol{E}_d)\sim \text{HashPRG}}[i^* = i \text{ and } \boldsymbol{E}_{i^*}^{-1/p}|x_{i^*}| \geq (1+\varepsilon)^{-1/p} \max_{i' \neq i} \boldsymbol{E}_i^{-1/p}|x_i|] \\
&\geq \frac{|x_i|^p}{(1+\varepsilon)\|x\|_p^p} - 1/\operatorname{poly}(d).
\end{aligned}$$

and with probability $\geq 1 - 1/\operatorname{poly}(d)$, $\sum_{i=1}^d (\boldsymbol{E}_i^{-1/p}|x_i|)^2 \leq d^{1-2/p}(\boldsymbol{E}_{i^*}^{-1/p}|x_{i^*}|)^2 \operatorname{polylog}(d)$. Let be a vector such that $\boldsymbol{f}_i = \boldsymbol{E}_i^{-1/p} x_i$. Condition on both the events. Then we have that the largest coordinate in $\boldsymbol{f}$ is at least a $1/(1+\varepsilon)$ factor larger than the second-largest coordinate and that $\|\boldsymbol{f}\|_2^2 = O(d^{1-2/p}\|\boldsymbol{f}\|_\infty^2)$. Now hashing the coordinates of $\boldsymbol{f}$ into a CountSketch data structure with $O(d^{1-2/p} \operatorname{polylog}(d)/\varepsilon^2)$ rows using $O(\log d)$ wise preserves the large coordinate of $\boldsymbol{f}$ using the analysis in [And17]. Using $O(1)$ independent CountSketch data structure and finding the coordinate $i \in [d]$ which hashes to the max bucket of the CountSketch data structure in each of the $O(1)$ repetitions, we can extract the coordinate i and output it as the ℓ_p sample.

Note that the update time is $O(1)$ as a block of random bits from HashPRG with parameter $k = d^c$ can be obtained in $O(1)$ time and then the time to evaluate the hash functions of the CountSketch data structure is $O(1)$ when using the constructions from Theorem 10.2.2. The overall space complexity of the data structure is $O(d^{1-2/p} \operatorname{polylog}(d)/\varepsilon^2)$ bits. However, we note that the final step of computing which $i \in [d]$ hashes to the max coordinate in all $O(1)$ copies of the CountSketch data structure takes $O(d)$ time.

10.5 Moment Estimation for $0 < p < 2$

We assume as usual that a vector $x \in \mathbb{R}^d$ is being maintained in the stream. The vector x is initialized to 0 and then receives m updates of the form $(i, v) \in [d] \times \{-M, \ldots, M\}$ where upon receiving (i, v), we update $x_i \leftarrow x_i + v$. We assume that both $m, M \leq \operatorname{poly}(d)$. At the end of the stream, we want to approximate $\|x\|_p^p$ up to a $1 \pm \varepsilon$ multiplicative factor. For ε such that $1/\sqrt{d} \leq \varepsilon \leq 1/d^c$ for a small enough constant, we show that the algorithm of [KNPW11] can be implemented in space of $O(\varepsilon^{-2}\log(d))$ *bits* of space and $O(\log d)$ update time per stream element. We measure the time complexity of the update algorithm in the Word RAM model with a word size of at least $\Omega(\log d)$.

We first give a high level overview of the moment estimation algorithm of [KNPW11]. Throughout the section, we assume that $1/\sqrt{d} \le \varepsilon \le 1/d^c$.

10.5.1 Overview of Moment Estimation Algorithm of [KNPW11]

Their 1-pass algorithm is based on the Geometric Mean estimator of Li [Li08]. Li gives an estimator to compute F_p moment of a vector using p-stable random variables. Let x be a fixed d dimensional vector and $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3 \in \mathbb{R}^d$ be random vectors with independent p-stable random variables. Then, Li showed that the estimator given by

$$\mathbf{Est} = \frac{(|\langle \boldsymbol{v}_1, x\rangle||\langle \boldsymbol{v}_2, x\rangle||\langle \boldsymbol{v}_3, x\rangle|)^{p/3}}{\left(\frac{2}{\pi}\Gamma\left(p/3\right)\Gamma\left(2/3\right)\sin\left(\pi p/6\right)\right)^3} \tag{10.2}$$

satisfies $\mathbf{E}_{\boldsymbol{v}_1,\boldsymbol{v}_2,\boldsymbol{v}_3}[\mathbf{Est}] = \|x\|_p^p$ and $\mathbf{Var}_{\boldsymbol{v}_1,\boldsymbol{v}_2,\boldsymbol{v}_3}[\mathbf{Est}] = O(\|x\|_p^{2p})$. A $1 \pm \varepsilon$ approximate estimator can be obtained by averaging $O(1/\varepsilon^2)$ independent copies of the estimator but it makes the update time $\Omega(\varepsilon^{-2})$ which is prohibitive. To decrease the variance of the estimator, [KNPW11] hash the coordinates of x into buckets and estimate the contribution of each bucket to the F_p moment of x separately. When there are heavy coordinates in the vector x, variance of the estimator may still be too large. Therefore, they estimate the contribution of the *heavy* elements using a different data structure they call HighEnd and use the Li's estimator to only estimate the contribution of the light elements using a data structure they call LightEstimator. We show that when $(1/\sqrt{d}) \le \varepsilon \le 1/d^c$ for a constant $c < 1/2$, we can implement their algorithm using $O(\varepsilon^{-2}\log d)$ bits of space and an update time of $O(\log d)$ per each element in the stream in the Word RAM model.

10.5.2 The HighEnd Data Structure

As described above, the algorithm of [KNPW11] estimates the F_p moment of heavy entries and light entries separately. Their heavy entry moment estimation method, at the end of the stream, takes in $L \subseteq [d]$ satisfying the following conditions:

1. $L \supseteq \{i \in [d] \mid |x_i|^p \ge \alpha\|x\|_p^p\}$,
2. if $i \in L$, then $|x_i|^p \ge (\alpha/C)\|x\|_p^p$ for a constant $C \ge 1$, and
3. we know the sign of x_i for all $i \in L$.

We show in Appendix C.2 how a set L satisfying the above properties can be computed in a turnstile stream using $O(\alpha^{-1}\log^2 d)$ bits of space and $O(\log d)$ update time per stream element. We use the CountSketch data structure [CCF04] along with the ExpanderSketch data structure [JST11] to obtain L. Note that the set L has size at most $O(1/\alpha)$.

Now we state the guarantees of the HighEnd data structure. We first define BasicHighEnd data structure and then define HighEnd by taking independent copies of the BasicHighEnd data structure.

Let α be such that $1/\alpha = O(1/\varepsilon^2)$ and let $s = \Theta(1/\alpha)$ be a large enough power of 2. Let $\boldsymbol{h} : [d] \to [s]$ is picked at random from an r_h-wise independent hash family for $r_h = \Theta(\log 1/\alpha)$. Let $r = \Theta(\log 1/\varepsilon)$ be a sufficiently large power of 2. Let $\boldsymbol{g} : [d] \to [r]$ be drawn at random from an r_g-wise independent hash family for $r_g = r$. For each $i \in [d]$, we associate a random complex root of unity given by $\exp(\mathrm{i}2\pi\boldsymbol{g}(i)/r)$ where i denotes $\sqrt{-1}$. We initialize s counters $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s$ to 0. Given an update of the form (i, v), we set $\boldsymbol{b}_{\boldsymbol{h}(i)} \leftarrow \boldsymbol{b}_{\boldsymbol{h}(i)} + \exp(\mathrm{i}2\pi\boldsymbol{g}(i)/r)v$.

The HighEnd data structure is defined by taking T independent copies of BasicHighEnd data structure with $T = O(\max(\log(1/\varepsilon), \log(1/\alpha))) = O(\log d)$. Each of the copies of the BasicHighEnd data structure is updated upon receiving an update (i, v) in the stream. Let $(\boldsymbol{h}^1, \boldsymbol{g}^1), \ldots, (\boldsymbol{h}^T, \boldsymbol{g}^T)$ be the hash functions corresponding to each of the BasicHighEnd data structures.

It is argued in [KNPW11] that storing the coefficients of complex numbers up to a precision of $\Theta(\log(d))$ bits suffices if the number of updates is $\mathrm{poly}(d)$, the magnitude of each update is bounded by $\mathrm{poly}(d)$ and $1/\sqrt{d} \le \varepsilon \le 1/d^c$. Thus the space complexity of HighEnd data structure (excluding the space required for storing the hash functions) is

$$O\left(\alpha^{-1}(\log d)^2\right) \text{ bits.}$$

By Theorem 10.2.2, for t a large enough constant we can construct random hash families $\mathcal{H} = \{ h : [d] \to [s] \}$ and $\mathcal{G} = \{ g : [d] \to [r] \}$ such that with probability $\ge 1 - 1/\mathrm{poly}(d)$, the hash families $\mathcal{H}$ and $\mathcal{G}$ are r_h and r_g wise independent respectively. Now, if $\boldsymbol{h}^1, \ldots, \boldsymbol{h}^T$ are sampled independently from $\mathcal{H}$ and $\boldsymbol{g}^1, \ldots, \boldsymbol{g}^T$ are sampled independently from $\mathcal{G}$, they can be evaluated on any input in $O(1)$ time and therefore the update time per each stream element is $O(T) = O(\log d)$ in the Word RAM model. Each hash function $\boldsymbol{h}^i$ and $\boldsymbol{g}^i$ can be stored using $O(d^{c/2})$ bits and therefore the space required to store the hash functions necessary for the HighEnd data structure is $o(d^c) = o(\varepsilon^{-2})$ bits.

At the end of stream, by Theorem 11 of [KNPW11] we can use the HighEnd data structure to compute a value Ψ such that with probability $\ge 7/8$,

$$|\Psi - \|x_L\|_p^p| \le O(\varepsilon)\|x\|_p^p.$$

We then have the following lemma:

Lemma 10.5.1. *Given $1/\sqrt{d} \le \varepsilon \le 1/d^c$ for a small enough constant, and α such that $1/\alpha = O(1/\varepsilon^2)$, there is a streaming algorithm that takes $O(\alpha^{-1}\log^2(d))$ bits of space and has an update time of $O(\log d)$ per stream element in the Word RAM model satisfying:*

1. *The algorithm outputs a set $L \subseteq [d]$ satisfying all the three properties stated above with probability $\ge 9/10$.*
2. *Conditioned on the list L satisfying those properties, the algorithm outputs a value Ψ such that with*

probability $\geq 65/100$,

$$|\Psi - \|x_L\|_p^p| \leq O(\varepsilon)\|x\|_p^p.$$

By taking median of Ψ output by $O(1)$ independent instances of the **HighEnd** *data structure, we have that conditioned on L satisfying all the properties, we have an estimate of $\|x_L\|_p^p$ with an additive error of $O(\varepsilon)\|x\|_p^p$ with probability $\geq 99/100$. Hence, by a union bound with probability $\geq 8/10$, the algorithm outputs both a good list L and a value Ψ satisfying $|\Psi - \|x_L\|_p^p| \leq O(\varepsilon)\|x\|_p^p$.*

Note that for $\alpha = \varepsilon^2\log(d)$, the algorithm uses $O(\varepsilon^{-2}\log d)$ *bits* of space. For this setting of α, we will now use the LightEstimator data structure of [KNPW11] to estimate $\|x_{[d]\setminus L}\|_p^p$. We will now describe the LightEstimator data structure and how it can be derandomized using HashPRG.

10.5.3 The LightEstimator Data Structure

Algorithm 10.3: LightEstimator using Independent p-stable random variables

```
Input: A parameter $p \in (0,2)$, accuracy parameter $\varepsilon$ such that $1/\sqrt{d} \leq \varepsilon \leq 1/d^c$ for a small
       enough constant $c$, a parameter $\alpha$ such that $1/\alpha = O(1/\varepsilon^2)$, a stream of updates
       $(i_1, v_1), \ldots, (i_m, v_m) \in [d] \times \{-M, \ldots, M\}$, a list $L \subseteq [d]$ of heavy coordinates
       revealed at the end of the stream
Output: An estimate of $\|x_{[d]\setminus L}\|_p^p$
$s \leftarrow O(1/\alpha)$;
$\boldsymbol{h} \leftarrow$ A hash function sampled from the construction in Theorem 10.2.3;
For $b \in [s]$ and $[j] \in [3]$, initialize $\boldsymbol{B}_{b,j} \leftarrow 0$;
For $i \in [d]$ and $[j] \in [3]$, let $\boldsymbol{A}_{i,j}$ be an independent $p$-stable random variable;
for $j = 1, \ldots, m$ do
    $\boldsymbol{B}_{\boldsymbol{h}(i_j),1} \leftarrow \boldsymbol{B}_{\boldsymbol{h}(i_j),1} + \boldsymbol{A}_{i_j,1}v_j$;
    $\boldsymbol{B}_{\boldsymbol{h}(i_j),2} \leftarrow \boldsymbol{B}_{\boldsymbol{h}(i_j),2} + \boldsymbol{A}_{i_j,2}v_j$;
    $\boldsymbol{B}_{\boldsymbol{h}(i_j),3} \leftarrow \boldsymbol{B}_{\boldsymbol{h}(i_j),3} + \boldsymbol{A}_{i_j,3}v_j$;
end
// The set $L$ is revealed to the algorithm
for $b = 1, \ldots, s$ do
    if $b \notin \boldsymbol{h}(L)$ then
        Est$(b) \leftarrow \frac{(|\boldsymbol{B}_{b,1}|\cdot|\boldsymbol{B}_{b,2}|\cdot|\boldsymbol{B}_{b,3}|)^{p/3}}{((2/\pi)\Gamma(p/3)\Gamma(2/3)\sin(\pi p/6))^3}$;
    else
        Est$(j) \leftarrow 0$;
    end
end
$\Phi \leftarrow \frac{s}{s-|\boldsymbol{h}(L)|}\sum_{b=1}^{s}$ Est$(b)$;
return $\Phi$;
```

As seen previously, the HighEnd data structure lets us compute the F_p moment of all the elements from L. We use the LightEstimator data structure to approximate the F_p moment of all the *light* elements i.e., the coordinates of x not in L.

Assume that at the end of processing the stream we are given a set $L \subseteq [d]$, $|L| \le 2/\alpha$ and for all $i \notin L$, $|x_i|^p < \alpha \|x\|_p^p$.

Let $s = \Theta(1/\alpha) \ge 10|L|$ be a large enough power of 2. For $i \in [d]$ and $j \in [3]$, let $\boldsymbol{A}_{i,j}$ denote an independent p-stable random variable. Let $\boldsymbol{h} : [d] \to [s]$ be a hash function drawn using the construction of Theorem 10.2.3 with parameters $z = \lceil 2/\alpha + 2 \rceil$ and a large enough constant C. For $i \in [s]$ and $j \in [3]$, initialize the counters $\boldsymbol{B}_{i,j} = 0$. On receiving an update (i, v) in the stream, for each $j \in [3]$, update $\boldsymbol{B}_{\boldsymbol{h}(i),j} \leftarrow \boldsymbol{B}_{\boldsymbol{h}(i),j} + \boldsymbol{A}_{i,j} \cdot v$. As argued in [KNPW11], we need to store the values $\boldsymbol{B}_{i,j}$ only up to a precision of $\Theta(\log d)$ bits. Hence, the space complexity of the LightEstimator data structure (excluding the space required to store/generate $\boldsymbol{A}_{i,j}$) is $O(\alpha^{-1} \log d)$ bits.

At the end of the stream, we receive the set $L \subseteq [d]$ of heavy hitters as specified in the previous section, and we will then compute an estimator for $\|x_{[d]\setminus L}\|_p^p$ as follows: for each $b \in [s] \setminus \boldsymbol{h}(L)$, define

$$\mathbf{Est}(b) \coloneqq \frac{(|\boldsymbol{B}_{b,1}| \cdot |\boldsymbol{B}_{b,2}| \cdot |\boldsymbol{B}_{b,3}|)^{p/3}}{\left(\frac{2}{\pi}\Gamma\left(p/3\right)\Gamma\left(2/3\right)\sin\left(\pi p/6\right)\right)^3}.$$

Let $\boldsymbol{h}(L) = \{\boldsymbol{h}(i) \mid i \in L\}$ denote the buckets to which the elements of L are hashed into. Define the following estimator

$$\Phi = \frac{s}{s - |\boldsymbol{h}(L)|} \sum_{b \in [s] \setminus \boldsymbol{h}(L)} \mathbf{Est}(b).$$

It is shown in [NW10] for $p = 1$ and extended to all $0 < p < 2$ in [KNPW11] that

$$\mathbf{E}_{\boldsymbol{h},\boldsymbol{A}}[\Phi] = (1 \pm \alpha^{10}) \|x_{[d]\setminus L}\|_p^p. \tag{10.3}$$

We extend their analysis and show an upper bound on $\mathbf{Var}_{\boldsymbol{h},\boldsymbol{A}}[\Phi]$.

Remark 10.5.2. Lemma 7 in [NW10] and Theorem 15 in [KNPW11] state that

$$\mathbf{E}_{\boldsymbol{h},\boldsymbol{A}}[\Phi] = (1 \pm O(\varepsilon)) \|x_{[d]\setminus L}\|_p^p$$

for $p = 1$ and $0 < p < 2$ respectively. It can be seen from the proof of Lemma 7 in [NW10], we can obtain the above stronger result by just picking C large enough when constructing the hash family $\mathcal{H}$ using the construction in Theorem 10.2.3. A similar argument works to extend for $0 < p < 2$.

Analysis of the Estimator

Claim 10.5.3.

$$\mathbf{Var}_{\boldsymbol{h},\boldsymbol{A}}[\Phi] \le O(\alpha)\|x\|_p^{2p}.$$

Proof. For simplicity, we define $\mathbf{Est}(b) = 0$ for all $b \in \boldsymbol{h}(L)$. We have

$$\begin{aligned}\mathbf{E}_{\boldsymbol{h},\boldsymbol{A}}[\Phi^2] &= \mathbf{E}_{\boldsymbol{h},\boldsymbol{A}}\left[\left(\frac{s}{s-|\boldsymbol{h}(L)|}\sum_{b=1}^{s}\mathbf{Est}(b)\right)^2\right] \\ &= \mathbf{E}_{\boldsymbol{h}}\left[\left(\frac{s}{s-|\boldsymbol{h}(L)|}\right)^2\left(\sum_{b=1}^{s}\mathbf{E}_{\boldsymbol{A}|\boldsymbol{h}}(\mathbf{Est}(b))^2 + \sum_{b\ne b'}\mathbf{E}_{\boldsymbol{A}|\boldsymbol{h}}(\mathbf{Est}(b))(\mathbf{Est}(b'))\right)\right]\end{aligned}$$

Now, for $b \notin \boldsymbol{h}(L)$, using the variance and mean of Li's estimator (10.2),

$$\mathbf{E}_{\boldsymbol{A}|\boldsymbol{h}}(\mathbf{Est}(b))^2 = \mathbf{Var}_{\boldsymbol{A}|\boldsymbol{h}}(\mathbf{Est}(b)) + (\mathbf{E}_{\boldsymbol{A}|\boldsymbol{h}}\,\mathbf{Est}(b))^2 = O(1)\left(\sum_{j:\boldsymbol{h}(j)=b}|x_j|^p\right)^2 \tag{10.4}$$

and as p-stable random variables denoted by $\boldsymbol{A}$ are independent, for $b \ne b'$ with $b, b' \notin \boldsymbol{h}(L)$,

$$\begin{aligned}\mathbf{E}_{\boldsymbol{A}|\boldsymbol{h}}(\mathbf{Est}(b))(\mathbf{Est}(b')) &= \mathbf{E}_{\boldsymbol{A}|\boldsymbol{h}}(\mathbf{Est}(b))\,\mathbf{E}_{\boldsymbol{A}|\boldsymbol{h}}(\mathbf{Est}(b')) \\ &= \sum_{j:\boldsymbol{h}(j)=b}|x_j|^p\sum_{j:\boldsymbol{h}(j)=b'}|x_j|^p.\end{aligned} \tag{10.5}$$

Hence,

$$\begin{aligned}\mathbf{E}_{\boldsymbol{h},\boldsymbol{A}}[\Phi^2] &= \mathbf{E}_{\boldsymbol{h}}\left[\left(\frac{s}{s-|\boldsymbol{h}(L)|}\right)^2\left(O(1)\sum_{b\notin\boldsymbol{h}(L)}(\sum_{i:\boldsymbol{h}(i)=b}|x_i|^p)^2 + \sum_{b\ne b':b,b'\notin\boldsymbol{h}(L)}(\sum_{i:\boldsymbol{h}(i)=b}|x_i|^p)(\sum_{i:\boldsymbol{h}(i)=b'}|x_i|^p)\right)\right] \\ &\le \mathbf{E}_{\boldsymbol{h}}\left[\left(\frac{s}{s-|\boldsymbol{h}(L)|}\right)^2\left(O(1)\sum_{b\notin\boldsymbol{h}(L)}(\sum_{i:\boldsymbol{h}(i)=b}|x_i|^p)^2 + \left(\sum_{i:\boldsymbol{h}(i)\notin\boldsymbol{h}(L)}|x_i|^p\right)^2\right)\right] \\ &\le O(1)\,\mathbf{E}_{\boldsymbol{h}}\left[\sum_{b\notin\boldsymbol{h}(L)}(\sum_{i:\boldsymbol{h}(i)=b}|x_i|^p)^2\right] + \mathbf{E}_{\boldsymbol{h}}\left[\sum_{i\ne i'}\left(\frac{s}{s-|\boldsymbol{h}(L)|}\right)^2\mathbf{1}[\boldsymbol{h}(i),\boldsymbol{h}(i')\notin\boldsymbol{h}(L)]|x_i|^p|x_{i'}|^p\right]\end{aligned}$$

where the last inequality follows from the fact that $|\boldsymbol{h}(L)| \le |L| \le s/10$. We first bound the second term. For any $i, i' \notin L$, with probability $1-\rho$, the hash function $\boldsymbol{h}$ is drawn from a hash family that is $|L|+2$-wise independent when restricted to the set $L \cup \{\, i, i' \,\}$ when restricted to $L \cup \{\, i, i' \,\}$. We can make $\rho \leq \varepsilon^C$ for any constant C by setting C large enough while sampling $\boldsymbol{h}$ from the hash family in Theorem 10.2.3. Let the event that the hash family is $|L| + 2$-wise independent with respect to $L \cup \{ i, i' \}$ be called Good. Conditioned on this event and the size $|\boldsymbol{h}(L)|$,

$$\mathbf{E}_{\boldsymbol{h}|\mathsf{Good},|\boldsymbol{h}(L)|}[\mathbf{1}[\boldsymbol{h}(i), \boldsymbol{h}(i') \notin \boldsymbol{h}(L)]] = \left(\frac{s - |\boldsymbol{h}(L)|}{s}\right)^2 \tag{10.6}$$

which gives

$$\begin{aligned}
&\mathbf{E}_{\boldsymbol{h}}\left[\sum_{i \neq i'} \left(\frac{s}{s - |\boldsymbol{h}(L)|}\right)^2 \mathbf{1}[\boldsymbol{h}(i), \boldsymbol{h}(i') \notin \boldsymbol{h}(L)] |x_i|^p |x_{i'}|^p\right] \\
&\leq \mathbf{E}_{\boldsymbol{h}|\mathsf{Good}}\left[\sum_{i \neq i'} \left(\frac{s}{s - |\boldsymbol{h}(L)|}\right)^2 \mathbf{1}[\boldsymbol{h}(i), \boldsymbol{h}(i') \notin \boldsymbol{h}(L)] |x_i|^p |x_{i'}|^p\right] \\
&\quad + \mathbf{E}_{\boldsymbol{h}|\neg\mathsf{Good}}\left[\sum_{i \neq i'} \left(\frac{s}{s - |\boldsymbol{h}(L)|}\right)^2 \mathbf{1}[\boldsymbol{h}(i), \boldsymbol{h}(i') \notin \boldsymbol{h}(L)] |x_i|^p |x_{i'}|^p\right] \times \mathbf{Pr}_{\boldsymbol{h}}[\neg\mathsf{Good}] \\
&\leq \|x_{[d]\setminus L}\|_p^{2p} + 2\|x_{[d]\setminus L}\|_p^{2p}\rho = (1 + 2\rho)\|x_{[d]\setminus L}\|_p^{2p}.
\end{aligned} \tag{10.7}$$

Here we used (10.6) to cancel out the $s^2/(s - |\boldsymbol{h}(L)|)^2$ factor in the expectation, and we used that $s/(s - |\boldsymbol{h}(L)|) \leq 10/9$ with probability 1 as $|\boldsymbol{h}(L)| \leq |L| \leq s/10$. Thus,

$$\begin{aligned}
\mathbf{E}_{\boldsymbol{h},A}[\Phi^2] \leq O(1)\, \mathbf{E}_{\boldsymbol{h}}\left[\sum_{b \notin h(L)} (\sum_{i : h(i) = b} |x_j|^p)^2\right] \\
+ (1 + 2\rho)\|x_{[d]\setminus L}\|_p^{2p}.
\end{aligned} \tag{10.8}$$

Now, we bound the first term.

$$\begin{aligned}
\mathbf{E}_{\boldsymbol{h}}\left[\sum_{b \notin \boldsymbol{h}(L)} \left(\sum_{i : \boldsymbol{h}(i) = b} |x_j|^p\right)^2\right] &= \mathbf{E}_{\boldsymbol{h}}\left[\sum_{b \notin \boldsymbol{h}(L)} \sum_{i : \boldsymbol{h}(i) = b} |x_i|^{2p} + \sum_{b \notin \boldsymbol{h}(L)} \sum_{i \neq i' : \boldsymbol{h}(i) = \boldsymbol{h}(i') = b} |x_i|^p |x_{i'}|^p\right] \\
&\leq \|x_{[d]\setminus L}\|_{2p}^{2p} + \mathbf{E}_{\boldsymbol{h}}\left[\sum_b \sum_{i \neq i'} \mathbf{1}[b \notin \boldsymbol{h}(L), \boldsymbol{h}(i) = \boldsymbol{h}(i') = b] |x_i|^p |x_{i'}|^p\right].
\end{aligned}$$

Again, for $i, i' \notin L$, with probability $1 - \rho$, the hash function $\boldsymbol{h}$ is drawn from a hash family that is $|L| + 2$-wise independent on the set $L \cup \{ i, i' \}$. Conditioning on that event, for any $b \in [s]$,

$$\mathbf{Pr}[\mathbf{1}[b \notin \boldsymbol{h}(L), \boldsymbol{h}(j) = b, \boldsymbol{h}(j') = b]] \leq O(1/s^2).$$

Hence, using an argument similar to the one in proving (10.7), we have

$$\mathbf{E}_{\boldsymbol{h}}\left[\sum_{b \notin \boldsymbol{h}(L)} \left(\sum_{j:\boldsymbol{h}(j)=b} |x_j|^p\right)^2\right]$$
$$= \|x_{[d]\setminus L}\|_{2p}^{2p} + O(1/s + \rho)\|x_{[d]\setminus L}\|_p^{2p}.$$

Therefore,

$$\begin{aligned}\mathbf{Var}_{\boldsymbol{h},\boldsymbol{A}}[\Phi] &\le O(1)\|x_{[d]\setminus L}\|_{2p}^{2p} + O(1/s+\rho)\|x_{[d]\setminus L}\|_p^{2p} + (1+O(\rho))\|x_{[d]\setminus L}\|_p^{2p} - (1-\alpha^{10})^2\|x_{[d]\setminus L}\|_p^{2p} \\ &\le O(1)\|x_{[d]\setminus L}\|_{2p}^{2p} + O(1/s+\rho)\|x_{[d]\setminus L}\|_p^{2p} + O(\rho+\alpha^{10})\|x_{[d]\setminus L}\|_p^{2p}.\end{aligned}$$

Now, $\|x_{[d]\setminus L}\|_{2p}^{2p} \le \alpha\|x\|_p^p\|x_{[d]\setminus L}\|_p^p$ using the fact that all coordinates in $x_{[d]\setminus L}$ have pth power at most $\alpha\|x\|_p^p$. As $\rho \le \varepsilon^C$ and $s = \Theta(1/\alpha)$,

$$\mathbf{Var}_{\boldsymbol{h},\boldsymbol{A}}[\Phi] \le O(\alpha)\|x\|_p^{2p}.$$ □

Let $\Phi_1, \ldots, \Phi_T$ be T independent copies of the estimator Φ for $T = \Theta(\log(1/\varepsilon)) = \Theta(\log d)$. As $\alpha = \varepsilon^2\log(d)$, $\mathbf{Var}[(\Phi_1 + \cdots + \Phi_T)/T] \le O(\varepsilon^2\log(d)/T)\|x\|_p^{2p} \le (\varepsilon^2/100)\|x\|_p^{2p}$. By Chebyshev inequality, with probability $\ge 99/100$, $\bar{\Phi} \coloneqq \frac{\Phi_1+\cdots+\Phi_T}{T}$ lies in the interval

$$[(1-\alpha^{10})\|x_{[d]\setminus L}\|_p^p - (\varepsilon/10)\|x\|_p^p, (1+\alpha^{10})\|x_{[d]\setminus L}\|_p^p + (\varepsilon/10)\|x\|_p^p].$$

Hence, using $O(\log d)$ independent instantiations of the LightEstimator data structure, we can obtain an estimate $\bar{\Phi}$ for $\|x_{[d]\setminus L}\|_p$ so that with probability $\ge 7/10$, $\Psi + \bar{\Phi} \in [(1-O(\varepsilon))\|x\|_p^p, (1+O(\varepsilon))\|x\|_p^p]$. Throughout the analysis we assumed that the p-stable random variables $\boldsymbol{A}_{i,j}$ are independent. To implement the algorithm in sublinear space, we derandomize p-stable random variables using HashPRG.

Derandomizing p-stable random variables using HashPRG

Lemma 10.5.4. *Given $p \in (0,2)$, an accuracy parameter ε such that $1/\sqrt{d} \le \varepsilon \le 1/d^c$ for a constant c, a parameter α such that $1/\alpha \le O(1/\varepsilon^2)$ and a stream of updates $(i_1, v_1), \ldots (i_m, v_m) \in [d]\times\{-M, \ldots, M\}$ for $m, M \le \mathrm{poly}(d)$ to the vector x, there is a streaming algorithm that uses $O(\alpha^{-1}\log d)$ bits of space and an update time of $O(\log d)$ per stream element in the Word RAM model. At the end of processing the stream, the algorithm takes in a set $L \subseteq [d]$ of heavy coordinates satisfying for all $i \notin L$, $|x_i|^p \le \alpha\|x\|_p^p$ and outputs a value Φ satisfying*

$$\mathbf{E}[\Phi] = (1 \pm \varepsilon^C)\|x_{[d]\setminus L}\|_p^p \pm \frac{1}{\mathrm{poly}(d)},$$

and

$$\mathbf{Var}[\Phi] \le O(\alpha)\|x\|_p^p.$$

The algorithm in above lemma is given by a modified version of Algorithm 10.3. Instead of using independent p-stable random variables $\boldsymbol{A}_{i,j}$, the algorithm uses HashPRG to obtain a pseudorandom string and uses the pseudorandom bits to compute the p-stable random variables.

Proof. Consider one instance of LightEstimator data structure as in Algorithm 10.3. It consists of the following objects:

1. A hash function $\boldsymbol{h} : [d] \to [s]$ for $s = O(1/\alpha)$ drawn from a hash family as stated in Theorem 10.2.3 with parameter $z = O(1/\alpha)$ and C being a large enough constant.
2. There is a table of $s = O(1/\alpha)$ counters each maintained with a precision of $O(\log d)$ bits.
3. The p-stable random variables $\boldsymbol{A}_{i,j}$ for $i \in [d]$ and $j = 1, 2, 3$.

The hash function $\boldsymbol{h}$ can be stored using $O(\alpha^{-1} \log d)$ bits by Theorem 10.2.3. The counters can be maintained using $O(\alpha^{-1} \log d)$ bits as well. So, we are left with derandomizing the p-stable random variables.

The algorithm overall needs $O(d \log d)$ uniform random bits to generate three p-stable random variables for each of the coordinates. We use HashPRG to obtain the pseudorandom bits and use them to generate p-stable random variables. We critically use the fact that our analysis of the estimator Φ as described in the previous section needs to use only the mean and variance of Φ to show that we only have to "fool" multiple $O(\log d)$ space algorithms and hence using HashPRG as described in Theorem 10.3.2 is enough to generate the pseudorandom bits to compute the p-stable random variables. Further, for each update in the stream, the necessary block of pseudorandom bits can be generated in $O(1)$ time for each update. The space required to store the randomness necessary for the pseudo random generator is $O(d^{\varepsilon}) = O(1/\varepsilon^2)$ bits when $\varepsilon \le 2c$ where $\varepsilon \le 1/d^c$.

So, the overall algorithm on each update (i, v) is as follows: We use $\boldsymbol{h}$ to hash i into one of the $O(1/\alpha)$ buckets. Note that $\boldsymbol{h}(i)$ can be computed in $O(1)$ time. Using the value of i, we generate a block of $O(\log d)$ pseudorandom bits from the pseudorandom generator and then use the bits to compute 3 samples from a p-stable distribution. Let the samples be $\boldsymbol{A}_{i,1}, \boldsymbol{A}_{i,2}, \boldsymbol{A}_{i,3}$. We update the counters $\boldsymbol{B}_{h(i),j} \leftarrow \boldsymbol{B}_{h(i),j} + \boldsymbol{A}_{i,j}v$ for $j \in [3]$. As discussed, generating the pseudorandom bits and updating the counters can be performed in $O(1)$ time (assuming the pseudorandom bits can be converted to samples from a p-stable distribution in $O(1)$ time).

Let $\boldsymbol{\gamma} \sim$ HashPRG be a string sampled from the pseudorandom generator. Let $\boldsymbol{A}_{\boldsymbol{\gamma}}$ denote the p-stable random variables generated using $\boldsymbol{\gamma}$. Let $\boldsymbol{A}$ denote p-stable random variables generated using a uniform random string of bits. Hence, the random variables $\boldsymbol{A}_{i,j}$ are independent. Fix a hash function $\boldsymbol{h}$, a vector x and a set of heavy elements L. Now consider the estimator we use to estimate the F_p moments of the light elements $[d] \setminus L$:

$$\Phi_{\boldsymbol{h},\boldsymbol{A}} = \frac{s}{(s - |\boldsymbol{h}(L)|)\theta} \times \sum_{b \in [s] \setminus \boldsymbol{h}(L)} (|\sum_{i:\boldsymbol{h}(i)=b} \boldsymbol{A}_{i,1}x_i| \cdot |\sum_{i:\boldsymbol{h}(i)=b} \boldsymbol{A}_{i,2}x_i| \cdot |\sum_{i:\boldsymbol{h}(i)=b} \boldsymbol{A}_{i,3}x_i|)^{p/3}$$

where we use θ to denote the denominator in (10.2). Fix some bucket $b \in [s] \setminus \boldsymbol{h}(L)$. Define

$$\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}} \coloneqq \frac{s}{(s - |\boldsymbol{h}(L)|)\theta}(|\sum_{i:\boldsymbol{h}(i)=b} \boldsymbol{A}_{i,1}x_i| \cdot |\sum_{i:\boldsymbol{h}(i)=b} \boldsymbol{A}_{i,2}x_i| \cdot |\sum_{i:\boldsymbol{h}(i)=b} \boldsymbol{A}_{i,3}x_i|)^{p/3}.$$

For $b \in \boldsymbol{h}(L)$, we define $\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}} = 0$. The quantity $\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}}$ can be computed by an $O(\log d)$ space algorithm in a single pass over the uniform random string of bits used to generate the p-stable random variables $\boldsymbol{A}$ by going over the values $i = 1, \ldots, d$ and ignoring the random bits that correspond to all i such that $\boldsymbol{h}(i) \neq b$. We formalize the algorithm by constructing an FSM $Q_{x,L,\boldsymbol{h},b}$ with $\text{poly}(d)$ states. Note that we fixed $x, L, \boldsymbol{h}$ and b. Let the state of automaton be of the form (i, c_1, c_2, c_3) where $i \in [d]$ and c_1, c_2, c_3 denote the counters. The FSM being in state (i, c_1, c_2, c_3) denotes that it has processed $x_1, \ldots, x_i$ and found that for $j \in [3]$

$$c_j = \sum_{i' \le i : h(i')=b} \boldsymbol{A}_{i',j}x_{i'}.$$

When in state (i, c_1, c_2, c_3), if $\boldsymbol{h}(i+1) \neq b$, the FSM directly transitions to the state $(i+1, c_1, c_2, c_3)$ ignoring the alphabet in the input to the FSM. If $\boldsymbol{h}(i) = b$, then the FSM reads the alphabet in the input string. Uses the $\{0, 1\}^{O(\log d)}$ size bit string that it reads to construct three p-stable random variables $\boldsymbol{A}_{i+1,1}, \boldsymbol{A}_{i+1,2}, \boldsymbol{A}_{i+1,3}$ and then transitions to the state $(i+1, c'_1, c'_2, c'_3)$ where for $j \in [3]$

$$c'_j = c_j + \boldsymbol{A}_{i+1,j}x_{i+1}.$$

Note that all the above operations are performed only with a precision of $O(\log d)$ bits. Hence, the Finite State Machine has only $\text{poly}(d)$ states. From the state (d, c_1, c_2, c_3), the algorithm transitions to (final, $(s/(s - |\boldsymbol{h}(L)|)\theta)(|c_1||c_2||c_3|)^{p/3}$).

Given a uniform random string as input, the final state of FSM $Q_{c,L,\boldsymbol{h},b}$ encodes the value $\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}}$ and given $\boldsymbol{\gamma} \sim \text{HashPRG}$ as input, the final state of FSM $Q_{c,L,\boldsymbol{h},b}$ encodes the value of $\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}_{\boldsymbol{\gamma}}}$.

Let $d^b_{\boldsymbol{h}}$ be the distribution of the value of $\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}}$ conditioned on $\boldsymbol{h}$. Now define $(d^b_{\boldsymbol{h}})'$ to be the distribution of $\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}_{\boldsymbol{\gamma}}}$ i.e., the value of the estimator for bth bucket computed using p-stable random variables generated from a random $\boldsymbol{\gamma}$ HashPRG. As FSM $Q_{x,L\boldsymbol{h},b}$ has only $\text{poly}(d)$ states, we obtain using Theorem 10.3.2 that

$$d_{\text{TV}}(d^b_{\boldsymbol{h}}, ((d_{\boldsymbol{h}})^b)') \le 1/\text{poly}(d).$$

The above is true for any fixing of $\boldsymbol{h}$, x, L and b. As for any values of $\boldsymbol{h}$ and $\boldsymbol{A}$, we have $|\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}}| \le \operatorname{poly}(d)$, we obtain that for any fixed $\boldsymbol{h}$, x, L and b,

$$|\mathbf{E}_{\boldsymbol{A}}[\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}}] - \mathbf{E}_{\boldsymbol{A}_\gamma}[\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}_M}]| \le \frac{1}{\operatorname{poly}(d)}$$

which implies that

$$|\mathbf{E}_{\boldsymbol{A}}[\Phi_{\boldsymbol{h},\boldsymbol{A}}] - \mathbf{E}_{\boldsymbol{A}_\gamma}[\Phi_{\boldsymbol{h},\boldsymbol{A}_\gamma}]| \le s \cdot \frac{1}{\operatorname{poly}(d)} \le \frac{1}{\operatorname{poly}(d)}.$$

Therefore, using (10.3) we have

$$\mathbf{E}_{\boldsymbol{h},\boldsymbol{A}_\gamma}[\Phi_{\boldsymbol{h},\boldsymbol{A}_\gamma}] = \mathbf{E}_{\boldsymbol{h},\boldsymbol{A}}[\Phi_{\boldsymbol{h},\boldsymbol{A}}] \pm 1/\operatorname{poly}(d) = (1 \pm \varepsilon^C)\|x_{[d]\setminus L}\|_p^p \pm 1/\operatorname{poly}(d). \tag{10.9}$$

Similarly, we have

$$(\Phi_{\boldsymbol{h},\boldsymbol{A}})^2 = \sum_{b,b' \in [s]} \Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}} \cdot \Phi^{(b')}_{\boldsymbol{h},\boldsymbol{A}}.$$

Again, for any fixed pair b, b', we can compute the product $\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}} \cdot \Phi^{(b')}_{\boldsymbol{h},\boldsymbol{A}}$ in $O(\log d)$ space using one pass over the uniform random string used to generate the p-stable random variables. We can construct a Finite State Machine very similar to the one above to show that the value $\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}} \cdot \Phi^{(b')}_{\boldsymbol{h},\boldsymbol{A}}$ can be computed by a machine with $\operatorname{poly}(d)$ states. Now, we have that the total variation distance between the distributions of the product when using a uniform random string to generate p-stable random variables and HashPRG to generate the p-stable random variables is at most $1/\operatorname{poly}(d)$ and hence we obtain that

$$|\mathbf{E}_{\boldsymbol{A}}(\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}} \cdot \Phi^{(b')}_{\boldsymbol{h},\boldsymbol{A}}) - \mathbf{E}_{\boldsymbol{A}_\gamma}(\Phi^{(b)}_{\boldsymbol{h},\boldsymbol{A}_\gamma} \cdot \Phi^{(b')}_{\boldsymbol{h},\boldsymbol{A}_\gamma})| \le 1/\operatorname{poly}(d).$$

Summing over all the pairs (b, b'), we obtain for any $x, L, \boldsymbol{h}$ that,

$$|\mathbf{E}_{\boldsymbol{A}}[(\Phi_{\boldsymbol{h},\boldsymbol{A}})^2] - \mathbf{E}_{\boldsymbol{A}_\gamma}[(\Phi_{\boldsymbol{h},\boldsymbol{A}_\gamma})^2]| \le \frac{1}{\operatorname{poly}(d)}$$

which implies

$$\mathbf{E}_{\boldsymbol{h},\boldsymbol{A}_\gamma}[(\Phi_{\boldsymbol{h},\boldsymbol{A}_\gamma})^2] \le \mathbf{E}_{\boldsymbol{h},\boldsymbol{A}}[(\Phi_{\boldsymbol{h},\boldsymbol{A}})^2] + \frac{1}{\operatorname{poly}(d)}.$$

We then have

$$\mathbf{Var}_{h,A_\gamma}[\Phi_{h,A_\gamma}] \le \mathbf{E}_{h,A}[(\Phi_{h,A})^2] + \frac{1}{\mathrm{poly}(d)} - (\mathbf{E}_{h,A}[\Phi_{h,A}])^2 + \frac{\mathbf{E}_{h,A}[\Phi_{h,A}]}{\mathrm{poly}(d)}.$$

As the $\mathrm{poly}(d)$ term can be made $\ge d^C$ for a large enough constant C, we obtain that

$$\mathbf{Var}_{h,A_\gamma}[\Phi_{h,A_M}] \le \mathbf{Var}_{h,A}[\Phi_{h,A}] + \frac{1}{\mathrm{poly}(d)} \le O(\alpha)\|x\|_p^{2p}$$

by Claim 10.5.3 and using the fact that x is a nonzero vector with integer coordinates and $\alpha \ge 1/\mathrm{poly}(d)$. □

10.5.4 Wrap-up

Theorem 10.5.5. *Given $p \in (0,2)$, an accuracy parameter ε such that $1/\sqrt{d} \le \varepsilon \le 1/d^c$ for a constant $0 < c < 1/2$ and a stream of updates $(i_1, v_1), \dots (i_m, v_m) \in [d] \times \{-M, \dots, M\}$ for $m, M \le \mathrm{poly}(d)$ to the vector x, there is a streaming algorithm that uses $O(\varepsilon^{-2}\log d)$ bits of space and has an update time of $O(\log d)$ per stream element that outputs with probability $\ge 7/10$, a value v such that*

$$v = (1 \pm \varepsilon)\|x\|_p^p.$$

Proof. Setting $\alpha = \varepsilon^2\log(d)$, the set of α heavy hitters L can be computed in $O(\alpha^{-1}\log^2(d)) = O(\varepsilon^{-2}\log d)$ bits of space using Lemma C.2.1 and has an update time of $O(\log d)$ per stream element. The set L satisfies all the properties in Lemma C.2.1 with probability $\ge 9/10$. By Lemma 10.5.1, the HighEnd data structure can be maintained in $O(\alpha^{-1}\log^2(d)) = O(\varepsilon^{-2}\log d)$ bits of space and has an update time of $O(\log d)$ per stream element. Conditioned on L satisfying all the properties, we have that the value Ψ output by HighEnd data structure satisfies with probability $\ge 9/10$,

$$\Psi = (1 \pm \varepsilon)\|x_L\|_p^p.$$

By Lemma 10.5.4, the LightEstimator data structure can be maintained in $O(\alpha^{-1}\log d) = O(\varepsilon^{-2})$ bits of space. We also have that the data structure can be update in $O(1)$ time per stream element and conditioned on the set L having all the properties, the value Φ output by the algorithm satisfies

$$\mathbf{E}[\Phi] = (1 \pm \varepsilon^C)\|x_{[d]\setminus L}\|_p^p + 1/\mathrm{poly}(d)$$

and

$$\mathbf{Var}[\Phi] = O(\varepsilon^2\log d)\|x\|_p^{2p}.$$

Maintaining $r = O(\log d)$ independent copies of LightEstimator in the stream and considering their outputs $\Phi_1, \ldots, \Phi_r$, conditioned on L having all the properties, we obtain using Chebyshev's inequality that with probability $\geq 99/100$,

$$\bar{\Phi} = \frac{\Phi_1 + \cdots + \Phi_r}{r} = (1 \pm \varepsilon^C)\|x_{[d]\setminus L}\|_p^p + \frac{1}{\mathrm{poly}(d)} + \varepsilon\|x\|_p^p.$$

Thus, by a union bound, with probability $\geq 7/10$,

$$\Psi + \bar{\Phi} = (1 \pm O(\varepsilon))\|x\|_p^p + \frac{1}{\mathrm{poly}(d)} = (1 \pm O(\varepsilon))\|x\|_p^p$$

using the fact that $\varepsilon > (1/\sqrt{d})$ and there is at least one nonzero integer coordinate in x. □

10.6 Derandomizing CountSketch with HashPRG

CountSketch [CCF04] is a random linear map of a vector $x \in \mathbb{R}^d$ to $Ax \in \mathbb{R}^D$. For parameters r, t such that $D = rt$, the CountSketch $\mathsf{CS}(x)$ is defined by two sequences of random independent hash functions: $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r : \{0, \ldots, d-1\} \to [t]$ and $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r : \{0, \ldots, d-1\} \to \{-1, +1\}$. To simplify our exposition we will assume that t is a power of two and that r is odd. For simplicity of presentation, in this section, we assume that the coordinates of x are 0-indexed so that $x = (x_0, \ldots, x_{d-1})$. Indexing $\mathsf{CS}(x) \in \mathbb{R}^D$ by $(i, j) \in [r] \times [t]$ the entries are defined as:

$$\mathsf{CS}(x)_{i,j} \coloneqq \sum_{\ell=0}^{d-1} \boldsymbol{s}_i(\ell)\, x_\ell\, [\boldsymbol{g}_i(\ell) = j].$$

For $x \in \mathbb{R}^d$ and $\ell \in \{0, \ldots, d-1\}$ we use $\mathsf{CS}(x)$ to approximate x_ℓ with the following estimator:

$$\hat{x}_\ell = \mathrm{median}(\{\boldsymbol{s}_i(\ell) \cdot \mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)} \mid i \in [r]\}) \ . \tag{10.10}$$

Charikar, Chen, and Farach-Colton [CCF04] upper bounded the estimation error $|\hat{x}_\ell - x_\ell|$ in terms of the norm of the vector x and the parameters r, t. Their analysis only relies on using *pairwise independent* hash functions (independently for each repetition), which require $O(r \log d)$ bits of storage and allow the estimator to be computed in $O(r)$ time assuming constant time arithmetic operations.

Minton and Price [MP14] presented a tighter analysis of the distribution of the estimation error, focusing on $r = \Theta(\log d)$ repetitions, under the assumption that the hash function values of $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r$ and $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r$ are *fully independent*. This assumption is used in order to argue about the Fourier transform of the error distribution. In our notation they show the following lemma:

Lemma 10.6.1. *For every $\alpha \in [0, 1]$, $\ell \in \{0, \ldots, d-1\}$,* $\mathbf{Pr}\left[|\hat{x}_\ell - x_\ell| > \alpha\,\Delta\right] < 2\exp\left(-\Omega\left(\alpha^2 r\right)\right)$*, where $\Delta = \|tail_t(x)\|_2/\sqrt{t}$.*

Literally storing fully random hash functions would require $O(rd\log t)$ bits, so this is not attractive when $d \gg t$, which is the setting where using CountSketch is of interest. Minton and Price note that for integer vectors $x \in \{-M,\ldots,+M\}^d$ where M is polynomial in d, it is possible to use the pseudorandom generator of Nisan [Nis92] to replace the fully independent hash functions, keeping the tail bound of Lemma 10.6.1 up to a $1/\mathrm{poly}(d)$ additive term. However, this comes with considerable overhead: The space complexity increases by an $\Omega(\log(dt))$ factor, and the time per update/query increases by a factor $\Omega(rt)$. Jayaram and Woodruff [JW18] later considered a modification of CountSketch (with the same space and error guarantees) and showed that the multiplicative space overhead can be reduced to $O((\log\log d)^2)$ using a pseudo-random generator for fooling halfspaces. The time complexity increases by an unspecified polylogarithmic factor compared to the fully random setting. Though it is technically not accurate we will still refer to their sketch as CountSketch.

In this section we present an alternative derandomization of [MP14] using hash functions computed using HashPRG. Specifically, for $i \in [r]$ and $\rho \in \{0,\ldots,d-1\}$ we use block number $(i-1)d+\rho$ from the output of HashPRG to get the random bits for $\boldsymbol{s}_i(\rho)$ and $\boldsymbol{g}_i(\rho)$. Since a given output block of HashPRG can be computed efficiently, these hash functions can be efficiently evaluated.

Theorem 10.6.2. *Let d be the dimension of the vectors and M be the maximum absolute value of coordinates in the vector. Let t and r be the parameters of the CountSketch map as defined above. Let $b \geq 2$ be an integer denoting the branching factor of HashPRG. Let $w = \Omega(\log d + \log M)$. There exists a randomized linear sketch $\mathsf{CS}_{HashPRG} : \{-M,\ldots,M\}^d \to \{-2^w,\ldots,2^w\}^{tr}$ that can be implemented on a word RAM with word size w with the following properties:*

- *The parameters required to define the map $\mathsf{CS}_{HashPRG}$ can be stored in $O(b\log_b d)$ words of space and given a vector $x \in \{-M,\ldots,M\}^d$, the resulting vector $\mathsf{CS}_{HashPRG}(x)$ can be stored in $O(rt)$ words of space.*
- *Given $\mathsf{CS}_{HashPRG}(x)$ and an update (ℓ, u_ℓ) corresponding to a vector u with a single nonzero entry u_ℓ, we can compute $\mathsf{CS}_{HashPRG}(x+u)$ in time $O(r\log_b d)$.*
- *For every $x \in \{-M,\ldots,M\}^d$, $\alpha \in [0,1]$, and $\ell \in \{0,\ldots,d-1\}$, we can compute an estimator $\hat{x}_\ell$ from $\mathsf{CS}_{HashPRG}$ in time $O(r\log_b d)$ such that $\mathbf{Pr}\left[|\hat{x}_\ell - x_\ell| > \alpha\,\Delta\right] < 2\exp\left(-\Omega\left(\alpha^2 r\right)\right) + 2^{-Cw}$, where $\Delta = \|tail_t(x)\|_2/\sqrt{t}$.*

Figure 10.1 compares Theorem 10.6.2 to previously known ways of choosing the hash functions for CountSketch. Our construction is the first one that is able to match the space usage of CountSketch with pairwise independent hash functions (for $r = O(\log d)$ repetitions) while showing the strong concentration known for fully random hash functions. With CountSketch table size $t = d^{\Omega(1)}$ and word length $w = O(\log d)$ we also match the update time of pairwise independence on the Word RAM.

Hash function	**Space in words**	**Bounds small error**	**Update time**
Pairwise independent [CCF04]	D	No	$\log d$
Fully random [MP14]	$d \log d$	Yes	$\log d$
Nisan's generator [MP14, Nis92]	$D \log(d)$	Yes	$t \log^3(d)$
Halfspace Fooling PRGs [JW18]	$D(\log\log d)^2$	Yes	$(\log d)^{O(1)}$
HashPRG ($b = t$)	D	Yes	$\log^2(d)/\log t$
HashPRG ($b = d^{\Omega(1)}$)	$D + b$	Yes	$\log d$

Table 10.1: Overview of CountSketch guarantees with different kinds of random hash functions. For simplicity, we focus on the case of $r = O(\log d)$ repetitions and d-dimensional input vectors that contain $O(\log d)$-bit integers such that the CountSketch itself (without hash functions) uses space $D = O(t \log d)$ words. With pairwise independence we can only tightly bound the probability of exceeding error $\Delta = \|\text{tail}_t(x)\|_2/\sqrt{t}$, while the other hash functions allow us to bound the probability of smaller errors. Time bounds are for implementation on a Word RAM with word size $w = O(\log d)$. Parameters with a particularly bad impact on space or time are highlighted in red color.

10.6.1 PRGs for Space-bounded Computation and CountSketch

Like Minton and Price [MP14] we will consider vectors $x \in \{-M, \ldots, M\}^d$ for a positive integer M. For concreteness, we consider CountSketch with entries that are w-bit machine words. We can relax the requirement from [MP14] that M is polynomial in d, and instead assume that $M < 2^{w-1}/d$, which is also necessary to ensure that there are no overflows when computing $\text{CS}(x)$.

To derandomize CountSketch, we describe a small-space algorithm for any fixed input vector x, query ℓ and threshold $\alpha\Delta$. The algorithm makes a single pass over the output from HashPRG and determines whether the estimator $\hat{x}_\ell$ computed using HashPRG has error exceeding $\alpha\Delta$. This is done *without* computing $\text{CS}(x)$, and in fact even without computing $\hat{x}_\ell$. The algorithm makes critical use of the *symmetry* of HashPRG, namely, that the distribution of hash values is unchanged by permuting the inputs using a mapping of the form $\rho \mapsto \rho \oplus \ell$. We stress that Nisan's generator does not have this symmetry property, and that we are not aware of an equally space-efficient finite state machine for evaluating the error of CountSketch using Nisan's generator.

The finite state machine. Consider $x \in \{-M, \ldots, M\}^d$, $\ell \in \{0, \ldots, d-1\}$, and a given error threshold $\alpha\Delta \in \mathbb{R}$. A choice of hash functions $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r : \{0, \ldots, d-1\} \to [t]$ and $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r : \{0, \ldots, d-1\} \to \{-1, +1\}$ can be represented as a binary string $\boldsymbol{\gamma} \in \{0,1\}^{rdw}$ where a block of w consecutive bits encodes hash values $\boldsymbol{s}_i(\rho)$ and $\boldsymbol{g}_i(\rho)$ for $i \in [r]$ and $\rho \in \{0, \ldots, d-1\}$. We order the blocks such that hash values with $i = 1$ come first, then $i = 2$ and so on. Concretely we may take $\boldsymbol{s}_i(\rho) = 2\boldsymbol{\gamma}_{(i-1)dw+\rho w} - 1$ and $\boldsymbol{g}_i(j) = 1 + \sum_{k=1}^{\log(t)} \boldsymbol{\gamma}_{(i-1)dw+(\rho+1)w-k} 2^{k-1}$ such that the hash values can be extracted from a block in constant time. We consider two ways of sampling the string $\boldsymbol{\gamma}$:

- First, we may choose $\boldsymbol{\gamma} \sim (U_2)^{rdw}$ with independent, random bits. It is easy to see that this is the same as choosing the hash functions with full independence, so Lemma 10.6.1 holds for this choice of $\boldsymbol{\gamma}$.

- Second, for every b, k such that $dr \leq b^k < 2^{cw}$ we can use HashPRG with block size $n = w$ to generate $\boldsymbol{\gamma} \sim G_k(*, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)$. This corresponds to the hash functions we use to derandomize CountSketch.

As we noted in the introduction, the distribution of $\boldsymbol{\gamma}^{\oplus \ell}$ is the same as the distribution of $\boldsymbol{\gamma}$. Thus, we can assume that an algorithm that makes a single pass over the string $\boldsymbol{\gamma}$ reads the rd blocks of bits in the order $0 \oplus \ell, 1 \oplus \ell, \ldots, (rd-1) \oplus \ell$ so that for each repetition $i \in [t]$, the algorithm gets to know the value $\boldsymbol{g}_i(\ell)$ before reading the pseudorandom bits corresponding to other blocks. Thus, for each repetition $i \in [t]$, we can assume that an algorithm making a single pass over the string $\boldsymbol{\gamma}$ *sees* the values $\boldsymbol{g}_i(0 \oplus \ell), \boldsymbol{g}_i(1 \oplus \ell), \ldots, \boldsymbol{g}_i((d-1) \oplus \ell)$ in that order and similarly the values of $\boldsymbol{s}_i$.

To analyze the properties of the CountSketch data structure constructed using the string $\boldsymbol{\gamma}$, we consider an FSM $Q = Q_{x,\ell,\alpha\Delta}$ with states

$$\{0, \ldots, r+1\}^3 \times \{-1, +1\} \times [t] \times \{0, \ldots, d\} \times \{-2^w, \ldots, 2^w\},$$

plus a special start state $\perp$. We use $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r$ and $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r$ to refer to the hash functions encoded by $\boldsymbol{\gamma}$. The idea is that the FSM Q computes the r simple estimators in (10.10) one at a time, and keeps track of the number of these estimators that deviate from x_ℓ by more than $\alpha\Delta$ (with separate accounting for overestimates and underestimates). More precisely, when Q is in state $(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7)$ it signifies that:

- It has fully processed the hash functions $\boldsymbol{s}_i$, $\boldsymbol{g}_i$ for $i < \beta_1$, that β_2 of these hash function pairs produced an estimate $\boldsymbol{s}_i(\ell) \cdot \mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)} < x_\ell - \alpha\Delta$, and β_3 pairs produced an estimate $\boldsymbol{s}_i(\ell) \cdot \mathsf{CS}(x)_{i,\boldsymbol{g}_i(\ell)} > x_\ell + \alpha\Delta$,
- $\boldsymbol{s}_{\beta_1}(\ell) = \beta_4$ and $\boldsymbol{g}_{\beta_1}(\ell) = \beta_5$, and
- $\beta_4 \sum_{0 \leq z < \beta_6} x_{z \oplus \ell} \boldsymbol{s}_{\beta_1}(z \oplus \ell)[\boldsymbol{g}_{\beta_1}(z \oplus \ell) = \boldsymbol{g}_{\beta_1}(\ell)] = \beta_7$.

From state $\perp$, the FSM Q transitions to $(1, 0, 0, \boldsymbol{s}_1(\ell), \boldsymbol{g}_1(\ell), 0, 0)$, where the values $\boldsymbol{s}_1(\ell), \boldsymbol{g}_1(\ell)$ are determined by the zeroth block since we assume that the FSM sees the blocks as they are ordered in the string $\boldsymbol{\gamma}^{\oplus \ell}$. From this point on, when in state $(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7)$:

- If $\beta_6 = d$ we have $\beta_7 = \boldsymbol{s}_{\beta_1}(\ell)\mathsf{CS}(x)_{\beta_1,\boldsymbol{g}_{\beta_1}(\ell)}$, so we can decide whether simple estimator number β_1 has an error above $\alpha\Delta$ and update β_2 or β_3 accordingly. Finally, we can increment β_1, set $\beta_6 = 0$, and update β_4, β_5 to reflect the values of the new hash values, available in the next block.
- Otherwise, when $\beta_1 \leq r$, Q has access to $\boldsymbol{s}_{\beta_1}(\beta_6 \oplus \ell)$ and $\boldsymbol{g}_{\beta_1}(\beta_6 \oplus \ell)$ from the next block of bits it reads. This allows us to increment β_6 when simultaneously increasing β_7 by $\beta_4 \boldsymbol{s}_{\beta_1}(\beta_6 \oplus \ell) x_{\beta_6 \oplus \ell}$ if $\boldsymbol{g}_{\beta_1}(\beta_6 \oplus \ell) = \beta_5$.
- Finally, when $\beta_1 = r + 1$ we ignore the rest of the input, remaining in the same state.

After reaching the end, the values β_2, β_3 determine whether the estimator $\hat{x}_\ell$ in (10.10) has an error of more than $\alpha\Delta$: If $\beta_2 \geq \lceil r/2 \rceil$ then $\hat{x}_\ell < x_\ell - \alpha\Delta$, if $\beta_3 \geq \lceil r/2 \rceil$ then $\hat{x}_\ell > x_\ell + \alpha\Delta$, and otherwise $|\hat{x}_\ell - x_\ell| \leq \alpha\Delta$.

The number of states in Q is $O(r^3 t d 2^w)$ and the number of blocks of bits read is rd, both of which are $2^{O(w)}$. Theorem 10.3.1 with $n = O(w)$ implies that the TV distance $\|Q(G_k(*, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)) - Q((U_n)^{2^k})\|$ is at most 2^{-cw} for *some* constant $c > 0$. The additive term 2^{-cw} can be made smaller than 2^{-Cw} for *any* constant $C > 1$ by adjusting the parameter n since a Word RAM with a constant factor larger word size can be simulated with a constant factor overhead in time. In particular, error probabilities grow by at most 2^{-Cw} when switching from fully random hash functions to hash functions defined by to HashPRG. Finally, note that the space usage of HashPRG is $O(w^2 b/\log b)$ bits, and that we can compute the hash function values $\boldsymbol{g}_1(\ell), \ldots, \boldsymbol{g}_r(\ell)$ and $\boldsymbol{s}_1(\ell), \ldots, \boldsymbol{s}_r(\ell)$ in time $O(rw/\log b)$.

10.6.2 Alternative Derandomizations of CountSketch with Nisan's PRG

We note that there are alternative derandomizations using Nisan's PRG that do not incur the naïve $O(\log d)$ factor overhead in terms of space in some regimes. We can use the fact that Nisan's PRG also "fools" small space algorithms that make multiple passes over the pseudorandom string. Specifically, Lemma 2.3 of [DPS11] shows that Nisan's PRG fools a small space algorithm that makes 2 passes over the pseudorandom string. We derandomize each repetition $i \in [r]$ separately. Fix an index $\ell \in [d]$ and repetition $i \in [r]$ and consider the string used to compute hash functions $\boldsymbol{g}_i$ and $\boldsymbol{s}_i$ for repetition i. An algorithm in the first pass over the string computes the bucket into which the index ℓ gets hashed into and in the second pass over the string computes the value, denoted by $\hat{x}_{i,\boldsymbol{g}_i(\ell)}$, of the bucket into which the ℓ-th coordinate gets hashed into in the i-th repetition. Finally, the algorithm terminates with the value $\boldsymbol{s}_i(\ell)\hat{x}_{i,\boldsymbol{g}_i(\ell)}$. As this algorithm overall takes $O(\log d)$ space, it is "fooled" by Nisan's PRG with a seed length of $O(\log^2 d)$. Thus, we obtain $\mathbf{Pr}[|\boldsymbol{s}_i(\ell)\hat{x}_{i,\boldsymbol{g}_i(\ell)} - x_\ell| \leq \alpha\Delta] \gtrsim \alpha$ as in proof of Theorem 4.1 of [MP14] even when each repetition of CountSketch is independently derandomized using a string sampled from Nisan's PRG. Overall, if each repetition is derandomized using an independent pseudorandom string, we obtain that $\mathbf{Pr}[|\hat{x}_\ell - x_\ell| > \alpha\Delta] \leq 2\exp(-\Omega(\alpha^2 r))$. While this derandomization has a fast update time, a drawback is that the overall seed length is $O(r \log^2 d)$ (a string of $O(\log^2 d)$ bits for each $i \in [r]$) which can be larger than the space complexity of CountSketch ($O(rt \log d)$ bits) when $t = o(\log d)$.

There is also another derandomization using Nisan's PRG which avoids space blow-up in the case of $rt = \omega(\log d)$. Instead of using independent samples from Nisan's PRG for each repetition of CountSketch, we derandomize the construction all-at-once. Consider the following algorithm that uses a single sample from Nisan's PRG to construct all hash functions $\boldsymbol{g}_i$ and $\boldsymbol{s}_i$. Fix a vector x and coordinate ℓ. The algorithm makes a first pass over the string to determine into which bucket the index ℓ gets hashed into in each of the repetitions. The information can be stored using $O(r \log t)$

bits. In the second pass over the random string, the algorithm can then determine if the estimate $\hat{x}_\ell$ satisfies $|\hat{x}_\ell - x_\ell| \le \alpha\Delta$. Overall the algorithm uses a space of $O(\log d + r \log t)$ bits. Using this fact, one can obtain a derandomization of CountSketch with the above estimation error guarantee and the CountSketch derandomized using Nisan's generator can be stored in $O(r \cdot t + \log d)$ words of space, which is asymptotically the same as the space complexity of CountSketch randomized using HashPRG. However, Nisan's generator needs to fool an $O(r \cdot \log t + \log d)$ space algorithm which makes the update time much slower, on a machine with $O(\log d)$ word size, compared to the derandomization using HashPRG which only has to fool an $O(\log d)$ space algorithm.

10.7 Private CountSketch

Pagh and Thorup [PT22] recently analyzed the estimation error of CountSketch data structure made private using the Gaussian Mechanism. Their analysis assumes $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r : \{0, \ldots, d-1\} \to [t]$ and $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r : \{0, \ldots, d-1\} \to \{+1, -1\}$ to be fully random. They define

$$\mathsf{PCS}(x) = \mathsf{CS}(x) + \boldsymbol{\nu}$$

where $\boldsymbol{\nu}$ is a $D = r \cdot t$ dimensional vector with independent Gaussian random variables of mean 0 and variance σ^2. By taking σ to be appropriately large, we obtain that $\mathsf{PCS}(x)$ is (ε, δ)-differentially private. For $\ell \in \{0, 1, \ldots, d-1\}$, we can define the estimator $\hat{x}_\ell$ as

$$\hat{x}_\ell = \mathrm{median}(\{\boldsymbol{s}_i(\ell) \cdot \mathsf{PCS}(x)_{i, \boldsymbol{g}_i(\ell)} \mid i \in [r]\}).$$

As we saw in Section 10.6, if σ were 0, then the hash functions $\boldsymbol{g}_i$ and $\boldsymbol{s}_i$ can be derandomized using HashPRG while obtaining tail bounds on the estimation error $|x_\ell - \hat{x}_\ell|$. A similar argument which crucially uses the symmetry property of HashPRG shows that PCS can also be derandomized using HashPRG. For the case of Private CountSketch with fully random hash functions the following theorem is shown in [PT22]:

Theorem 10.7.1 ([PT22])**.** *For every $\alpha \in [0, 1]$ and every $\ell \in \{0, \ldots, d-1\}$, the estimation error of private CountSketch with r repetitions, table size t and $\boldsymbol{\nu} \sim N(0, \sigma^2)^D$, then*

$$\mathbf{Pr}[|\hat{x}_\ell - x_\ell| \ge \alpha \max(\Delta, \sigma)] \le 2\exp(-\Omega(\alpha^2 r))$$

where $\Delta = \|tail_t(x)\|_2/\sqrt{t}$.

We now derandomize the requirement that the hash functions $\boldsymbol{g}_i$ and $\boldsymbol{s}_i$ be fully random. The proof is an extension of the proof in Section 10.6 and proceeds very similarly. We detail it below for completeness. Instead of constructing an FSM, we describe a small space algorithm which makes a single pass over the randomness while updating its state after reading a block of random bits. The algorithm can be easily converted to an FSM.

Fix a vector $x \in \mathbb{R}^d$ (corresponds to the final value of the vector in the stream), $v \in \mathbb{R}^D$ (corresponds to the Gaussian vector we add to CountSketch to make it private), a coordinate $\ell \in [d]$ and a parameter α. We initialize three variables $\mathsf{acc} = \mathsf{deficit} = \mathsf{excess} = 0$. The algorithm reads a block of w bits from the input string. Using the symmetry of HashPRG we again assume that the FSM sees the blocks of $\boldsymbol{\gamma}$ as they are in $\boldsymbol{\gamma}^{\oplus \ell}$. Thus, for each repetition, the block that the FSM sees first can be used to determine $\boldsymbol{g}_i(\ell)$ and $\boldsymbol{s}_i(\ell)$. For the first repetition, using the zeroth block of bits, the FSM computes and stores the values $\boldsymbol{g}_1(\ell)$ and $\boldsymbol{s}_1(\ell)$ locally and update the accumulator acc to $\boldsymbol{s}_1(\ell)x_{0\oplus\ell}$ (note that $x_{0\oplus\ell} = x_\ell$) and move to the next block of bits in the input string.

Suppose we are reading the j-th block of random bits. Again, we can determine $\boldsymbol{g}_i(j \oplus \ell)$ and $\boldsymbol{s}_i(j \oplus \ell)$. If $\boldsymbol{h}_1(j \oplus \ell) \neq \boldsymbol{g}_1(\ell)$, we move to reading the $(j+1)$-th block. If $\boldsymbol{g}_1(j \oplus \ell) = \boldsymbol{g}_1(\ell)$, we add $\boldsymbol{s}_1(j \oplus \ell)x_{j\oplus\ell}$ to the accumulator and move to the $(j+1)$-th block. After reading d blocks, if $\boldsymbol{s}_1(\ell)(\mathsf{acc}+v_{1,\boldsymbol{g}_1(\ell)}) \geq x_\ell + \alpha\max(\Delta, \sigma)$ we increase the variable excess by 1. If $\boldsymbol{s}_1(\ell)(\mathsf{acc}+v_{1,\boldsymbol{g}_1(\ell)}) \leq x_\ell - \alpha\max(\Delta, \sigma)$, we increase $\mathsf{deficit}$ by 1. Then we zero out the variable acc, remove the stored values $\boldsymbol{g}_1(\ell)$ and $\boldsymbol{s}_1(\ell)$ and repeat the process by reading the next block of bits. We again determine $\boldsymbol{g}_2(\ell)$ and $\boldsymbol{s}_2(\ell)$ and set acc to $\boldsymbol{s}_2(\ell)x_\ell$. We then move to the next block and so on. We do the process in total for r of times. Finally, if $\mathsf{excess} < r/2$ and $\mathsf{deficit} < r/2$, we set the variable Status to $\mathsf{Success}$ and otherwise set Status to $\mathsf{Failure}$.

The algorithm, at any point of time needs to store only $O(\log d)$ bits. As the algorithm needs only "read" access to x, the entire algorithm can be converted to an FSM, which we call $Q_{x,v,\ell,\alpha}$, that has $\text{poly}(d)$ states over the alphabet $\{0, 1\}^w$. The variable Status determines if the FSM ends in a state $\mathsf{Success}$ or $\mathsf{Failure}$.

If the input string to the FSM is uniformly random, then the hash functions $\boldsymbol{g}_i$ and $\boldsymbol{s}_i$ are fully random. Therefore, by Theorem 10.7.1, we obtain that

$$\mathbf{E}_v[\mathbf{Pr}_{\boldsymbol{\gamma}\sim U}[\text{Final State} = \mathsf{Success}]] \geq 1 - 2\exp(-\Omega(\alpha^2 r)).$$

Now consider HashPRG with a block size $w = \Omega(\log d)$. By Theorem 10.3.2, for any fixed x, v, ℓ and α that

$$\begin{aligned}&\mathbf{Pr}_{\boldsymbol{\gamma}\sim \text{HashPRG}}[\text{Final State of FSM } Q_{x,v,\ell,\alpha} \text{ on input } \boldsymbol{\gamma}]\\ &\quad \geq \mathbf{Pr}_{\boldsymbol{\gamma}\sim U}[\text{Final State of FSM } Q_{x,v,\ell,\alpha} \text{ on input } \boldsymbol{\gamma}] - O(2^{-cw}).\end{aligned}$$

By taking an expectation over $\boldsymbol{v} \sim N(0, \sigma^2)^D$, we have

$$\mathbf{E}_{\boldsymbol{v}}\,\mathbf{Pr}_{\boldsymbol{\gamma}\sim \text{HashPRG}}[\text{Final State of FSM } Q_{x,\boldsymbol{v},\ell,\alpha} \text{ on input } \boldsymbol{\gamma}] \geq 1 - 2\exp(-\Omega(\alpha^2 r)) - O(2^{-cw}).$$

We therefore obtain that with probability $\geq 1 - 2\exp(-\Omega(\alpha^2 r)) - 2 \cdot 2^{-cw}$ over $\boldsymbol{v} \sim N(0, \sigma^2)^D$ and $\boldsymbol{\gamma} \sim$ HashPRG, if $\boldsymbol{g}_i$ and $\boldsymbol{s}_i$ are hash functions constructed as a function of $\boldsymbol{\gamma}$ as described in the above algorithm, then with probability $\geq 1 - \exp(-\Omega(\alpha^2 r)) - O(2^{-cw})$ over $\boldsymbol{\nu}$ and $\boldsymbol{\gamma}$,

$$|\mathrm{median}_{i \in [r]}(\boldsymbol{s}_i(\ell)(\sum_{j=0}^{d-1}[\boldsymbol{g}_i(j) = \boldsymbol{g}_i(\ell)]\boldsymbol{s}_j(i)x_j + \nu_{i,\boldsymbol{g}_i(\ell)})) - x_\ell| \leq \alpha \max(\Delta, \sigma).$$

Thus, we have the following theorem.

Theorem 10.7.2. *For every $\alpha \in [0,1]$ and every $\ell \in \{0, \ldots, d-1\}$, the estimation error of private CountSketch with r repetitions, table size t derandomized using HashPRG with a block size of w and $\boldsymbol{\nu} \sim N(0, \sigma^2)^D$, then*

$$\mathbf{Pr}[|\hat{x}_\ell - x_\ell| \geq \alpha \max(\Delta, \sigma)] \leq 2\exp(-\Omega(\alpha^2 r)) + O(2^{-cw})$$

where $\Delta = \|tail_t(x)\|_2/\sqrt{t}$.

10.8 Estimating $\|x\|_\infty$

Let $x \in \mathbb{R}^d$ be the underlying vector we are maintaining in a turnstile stream. Assume that the coordinates of x are integers bounded in absolute value by $\mathrm{poly}(d)$. We give a simple algorithm that uses only $O(\varepsilon^{-2}\log d \log 1/\varepsilon)$ bits of space and approximates $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$. We give a matching lower bound and show that our algorithm is tight up to constant factors.

Let $t \leq d$ be a parameter that we set later. Let $\boldsymbol{L} : \mathbb{R}^d \to \mathbb{R}^t$ be a randomized linear map defined as

$$(\boldsymbol{L}x)_i = \sum_{j:\boldsymbol{h}(j)=i} \boldsymbol{s}(i)x_i.$$

Here we assume that the hash function $\boldsymbol{h}$ is drawn from a 2-wise independent hash family $\mathscr{H} = \{ h : [d] \to [t] \}$ and the sign function $\boldsymbol{s}$ is drawn from a 4-wise independent hash family $\mathcal{S} = \{ s : [d] \to \{+1, -1\} \}$. We note that there exist families $\mathscr{H}$ and $\mathcal{S}$ such that $\boldsymbol{h}$ and $\boldsymbol{s}$ can be sampled from their respective families and stored using $O(\log d)$ bits. We prove the following lemma:

Lemma 10.8.1. *Given a parameter α, if $t \geq 1/2\alpha^4\delta$, then with probability $\geq 1 - 3\delta$, the following simultaneously hold:*

1. $\|\boldsymbol{L}\|_\infty = \|x\|_\infty \pm (2\sqrt{\alpha}/\delta^{1/4})\|x\|_2$ *and*
2. $\|\boldsymbol{L}x\|_2^2 \leq (1 + 2\alpha^2)\|x\|_2^2$.

Proof. Define $\textsc{Large} \coloneqq \{j \in [d] \mid |x_j| \geq \alpha\|x\|_2\}$ and $\textsc{Small} \coloneqq [d] \setminus \textsc{Large}$. Note that $|\textsc{Large}| \leq 1/\alpha^2$. Let $x_{\textsc{Large}} \in \mathbb{R}^d$ be the d-dimensional vector with only the $\textsc{Large}$ coordinates of vector x and define $x_{\textsc{Small}} = x - x_{\textsc{Large}}$. The following result is now a simple consequence of the 2-wise independence of $\boldsymbol{h}$.

Lemma 10.8.2. *If $t \ge |\textsc{Large}|^2/(2\delta)$, then with probability $\ge 1-\delta$, for all $j, j' \in \textsc{Large}$ with $j \ne j'$, we have $\boldsymbol{h}(j) \ne \boldsymbol{h}(j')$.*

Proof. For $j, j' \in \textsc{Large}$ with $j \ne j'$, let $\boldsymbol{X}_{j,j'} = 1$ if $\boldsymbol{h}(j) = \boldsymbol{h}(j')$ and 0 otherwise. Using the 4-wise independence of $\mathscr{H}$, we have $\mathbf{E}[\boldsymbol{X}_{j,j'}] = 1/t$. Hence, $\mathbf{E}[\sum_{j<j'} \boldsymbol{X}_{j,j'}] \le |\textsc{Large}|^2/(2t)$. By Markov's inequality, with probability $\ge 1-\delta$, $\sum_{j<j'} \boldsymbol{X}_{j,j'} \le |\textsc{Large}|^2/(2t\delta) \le 1/2$ if $t \ge |\textsc{Large}|^2/\delta$. By definition, the random variable $\sum_{j<j'} \boldsymbol{X}_{j,j'}$ takes only non-negative integer values. Hence, we obtain that with probability $\ge 1-\delta$, $\sum_{j<j'} \boldsymbol{X}_{j,j'} = 0$ which implies that the hash function $\boldsymbol{h}$ hashes each of the coordinates in the set $\textsc{Large}$ to distinct locations. □

As $t \ge 1/2\alpha^4\delta \ge |\textsc{Large}|^2/2\delta$, we get that all the coordinates in the set $\textsc{Large}$ are hashed to distinct buckets by the hash function $\boldsymbol{h}$ with probability $1-\delta$. We now bound $\|\boldsymbol{L}x_{\textsc{Small}}\|_\infty$. By definition of the set $\textsc{Small}$, we have $\|x_{\textsc{Small}}\|_\infty \le \alpha\|x\|_2$ and $\|x_{\textsc{Small}}\|_2 \le \|x\|_2$. Let $i \in [t]$ be arbitrary. We have $(\boldsymbol{L}x_{\textsc{Small}})_i = \sum_{j\in\textsc{Small}} [\boldsymbol{h}(j) = i]\boldsymbol{s}(j)x_j$ and

$$\mathbf{E}[(\boldsymbol{L}x_{\textsc{Small}})_i^2] = \sum_{j,j'\in\textsc{Small}} \mathbf{Pr}[\boldsymbol{h}(j) = i, \boldsymbol{h}(j') = i]\, \mathbf{E}[\boldsymbol{s}(j)\boldsymbol{s}(j')]x_j x_{j'}$$

using the independence of $\boldsymbol{h}$ and $\boldsymbol{s}$. For $j \ne j' \in \textsc{Small}$, using the 4-wise independence of $\boldsymbol{s}$, we get $\mathbf{E}[\boldsymbol{s}(j)\boldsymbol{s}(j')] = 0$ and therefore, the above expression simplifies to

$$\mathbf{E}[(\boldsymbol{L}x_{\textsc{Small}})_i^2] = \sum_{j\in\textsc{Small}} \mathbf{Pr}[\boldsymbol{h}(j) = i]x_j^2 = \|x_{\textsc{Small}}\|_2^2/t.$$

Similarly, we have

$$\mathbf{E}[(\boldsymbol{L}x_{\textsc{Small}})_i^4] = \sum_{j_1,j_2,j_3,j_4\in\textsc{Small}} \mathbf{Pr}[\bigwedge_{k=1}^{4}(\boldsymbol{h}(j_k) = i)]\, \mathbf{E}[\prod_{k=1}^{4} \boldsymbol{s}(j_k)] \prod_{k=1}^{4} x_{j_k}.$$

We now see using the 4-wise independence of $\boldsymbol{s}$ that $\mathbf{E}[\boldsymbol{s}(j_1)\boldsymbol{s}(j_2)\boldsymbol{s}(j_3)\boldsymbol{s}(j_4)]$ is nonzero only when all the indices j_1, j_2, j_3, j_4 are equal, or we can pair the indices j_1, j_2, j_3, j_4 into two groups taking same values. Hence,

$$\begin{aligned}\mathbf{E}[(\boldsymbol{L}x_{\textsc{Small}})_i^4] &= \frac{\|x_{\textsc{Small}}\|_4^4}{t} + \frac{6}{t^2}\sum_{j_1<j_2\in\textsc{Small}} x_{j_1}^2 x_{j_2}^2 \\ &= \frac{\|x_{\textsc{Small}}\|_4^4}{t} + \frac{3}{t^2}\left(\|x_{\textsc{Small}}\|_2^4 - \|x_{\textsc{Small}}\|_4^4\right)\end{aligned}$$

which then implies

$$\mathbf{Var}[(\boldsymbol{L}x_{\textsc{Small}})_i^2] \le (1/t)\|x_{\textsc{Small}}\|_4^4 + (2/t^2)\|x_{\textsc{Small}}\|_2^4.$$

Since $\|x_{\text{SMALL}}\|_4^4 \leq \|x_{\text{SMALL}}\|_\infty^2 \|x_{\text{SMALL}}\|_2^2 \leq \alpha^2 \|x\|_2^4$, we get $\mathbf{Var}[(\boldsymbol{L}x_{\text{SMALL}})_i^2] \leq (2/t^2 + \alpha^2/t)\|x\|_2^4$. By Chebyshev's inequality,

$$\mathbf{Pr}[(\boldsymbol{L}x_{\text{SMALL}})_i^2 \geq \|x_{\text{SMALL}}\|_2^2/t + \gamma] \leq \frac{(2/t^2 + \alpha^2/t)\|x\|_2^4}{\gamma^2}.$$

By a union bound over all t values of i, we get that $\|\boldsymbol{L}x_{\text{SMALL}}\|_\infty^2 \leq \|x_{\text{SMALL}}\|_2^2/t + \gamma$ with probability $\geq 1 - (2/t + \alpha^2)\|x\|_2^4/\gamma^2$. For $t \geq 1/\alpha^2$ and $\gamma = (2\alpha/\sqrt{\delta})\|x\|_2^2$, we have that with probability $\geq 1 - \delta$, $\|\boldsymbol{L}x_{\text{SMALL}}\|_\infty^2 \leq (\alpha^2 + 2\alpha/\sqrt{\delta})\|x\|_2^2 \leq (3\alpha/\sqrt{\delta})\|x\|_2^2$. We thus finally have that if $t \geq 1/\alpha^2$, then with probability $\geq 1 - \delta$,

$$\|\boldsymbol{L}x_{\text{SMALL}}\|_\infty \leq \frac{2\sqrt{\alpha}}{\delta^{1/4}}\|x\|_2.$$

Hence, by a union bound, with probability $\geq 1 - 2\delta$, $\|\boldsymbol{L}x_{\text{LARGE}}\|_\infty = \|x_{\text{LARGE}}\|_\infty$ and $\|\boldsymbol{L}x_{\text{SMALL}}\|_\infty \leq (2\sqrt{\alpha}/\delta^{1/4})\|x\|_2$. Condition on this event. If $\|x\|_\infty \geq \alpha\|x\|_2$, then $\|x_{\text{LARGE}}\|_\infty = \|x\|_\infty$ and by triangle inequality, we get

$$\begin{aligned}
\|\boldsymbol{L}x\|_\infty &= \|\boldsymbol{L}x_{\text{LARGE}} + \boldsymbol{L}x_{\text{SMALL}}\|_\infty \\
&= \|\boldsymbol{L}x_{\text{LARGE}}\|_\infty \pm \|\boldsymbol{L}x_{\text{SMALL}}\|_\infty \\
&= \|x\|_\infty \pm (2\sqrt{\alpha}/\delta^{1/4})\|x\|_2.
\end{aligned}$$

If $\|x\|_\infty < \alpha\|x\|_2$, then $\text{LARGE} = \emptyset$ and $\|\boldsymbol{L}x\|_\infty = \|\boldsymbol{L}x_{\text{SMALL}}\|_\infty \leq (2\sqrt{\alpha}/\delta^{1/4})\|x\|_2$ which clearly satisfies $\|\boldsymbol{L}x\|_\infty = \|x\|_\infty \pm (2\sqrt{\alpha}/\delta^{1/4})\|x\|_2$.

We will now bound $\|\boldsymbol{L}x\|_2^2$. First we have,

$$\begin{aligned}
\|\boldsymbol{L}x\|_2^2 &= \sum_{i\in[t]} \left(\sum_{j\in[d]:\boldsymbol{h}(j)=i} \boldsymbol{s}(j)x_j \right)^2 \\
&= \sum_{i\in[t]} \left(\sum_{\substack{j\in[d] \\ \boldsymbol{h}(j)=i}} x_j^2 + \sum_{\substack{j_1\neq j_2\in[d]: \\ \boldsymbol{h}(j_1)=\boldsymbol{h}(j_2)=i}} \boldsymbol{s}(j_1)\boldsymbol{s}(j_2)x_{j_1}x_{j_2} \right) \\
&= \|x\|_2^2 + 2\sum_{i\in[t]} \sum_{j_1<j_2\in[d]} [\boldsymbol{h}(j_1) = \boldsymbol{h}(j_2) = i]\boldsymbol{s}(j_1)\boldsymbol{s}(j_2)x_{j_1}x_{j_2}.
\end{aligned}$$

By 4-wise independence of $\boldsymbol{s}$, we get $\mathbf{E}[\boldsymbol{s}(j_1)\boldsymbol{s}(j_2)] = 0$ for $j_1 \neq j_2$ and therefore get $\mathbf{E}[\|\boldsymbol{L}x\|_2^2] =$

$\|x\|_2^2$. We now bound $\mathbf{Var}(\|\boldsymbol{L}x\|_2^2)$.

$$\begin{aligned}\mathbf{Var}(\|\boldsymbol{L}x\|_2^2) &= \mathbf{E}[(\|\boldsymbol{L}x\|_2^2 - \|x\|_2^2)^2]\\ &= 4\sum_{i_1,i_2}\sum_{\substack{j_1<j_2\\ j_3<j_4}} \mathbf{Pr}[\boldsymbol{h}(j_1) = \boldsymbol{h}(j_2) = i_1, \boldsymbol{h}(j_3) = \boldsymbol{h}(j_4) = i_2] \cdot \mathbf{E}[\boldsymbol{s}(j_1)\boldsymbol{s}(j_2)\boldsymbol{s}(j_3)\boldsymbol{s}(j_4)]x_{j_1}x_{j_2}x_{j_3}x_{j_4}.\end{aligned}$$

Note that by 4-wise independence of the hash family from which the function $\boldsymbol{s}$ is drawn, we get that $\mathbf{E}[\boldsymbol{s}(j_1)\boldsymbol{s}(j_2)\boldsymbol{s}(j_3)\boldsymbol{s}(j_4)]$ is 0 unless $j_1 = j_3$ and $j_2 = j_4$ (since $j_1 < j_2$ and $j_3 < j_4$). If $j_1 = j_3$ and $j_2 = j_4$, we additionally have that $\mathbf{Pr}[\boldsymbol{h}(j_1) = \boldsymbol{h}(j_2) = i_1, \boldsymbol{h}(j_3) = \boldsymbol{h}(j_4) = i_2] = 0$ unless $i_1 = i_2$. Thus, the above expression simplifies to

$$\begin{aligned}\mathbf{Var}(\|\boldsymbol{L}x\|_2^2) &= 4\sum_{i}\sum_{j_1<j_2} \mathbf{Pr}[\boldsymbol{h}(j_1) = \boldsymbol{h}(j_2) = i]x_{j_1}^2 x_{j_2}^2\\ &= 4\sum_i (1/t^2)\sum_{j_1<j_2} x_{j_1}^2 x_{j_2}^2\\ &= (2/t)(\|x\|_2^4 - \|x\|_4^4).\end{aligned}$$

For $t \ge 1/2\alpha^4\delta$, we have $\mathbf{Var}(\|\boldsymbol{L}x\|_2^2) \le 4\alpha^4\delta\|x\|_2^4$ and by Chebyshev inequality, we get that $\mathbf{Pr}[\|\boldsymbol{L}x\|_2^2 \ge \|x\|_2^2 + 2\alpha^2\|x\|_2^2] \le \delta$. By the union bound, we obtain the result. □

We now prove the following theorem.

Theorem 10.8.3. *There is a turnstile stream algorithm using $O(\varepsilon^{-2}\log(1/\varepsilon)\log d)$ bits of space and outputs an estimate to $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$ with probability $\ge 9/10$.*

Proof. In the above lemma, setting $\delta = 1/100, \alpha = \varepsilon^2/160$, we get that for $t = C/\varepsilon^8$ for a large enough constant C, with probability $\ge 97/100$, $\|\boldsymbol{L}x\|_\infty = \|x\|_\infty \pm (\varepsilon/2)\|x\|_2$ and $\|\boldsymbol{L}x\|_2^2 \le (1 + 4\varepsilon^4)\|x\|_2^2$. Condition on this event. From [CCF04], if a vector $y \in \mathbb{R}^m$ is being updated in a turnstile stream, the CountSketch data structure with parameters $t = O(1/\varepsilon^2)$ and $r = O(\log m)$ can be used to recover a vector $\hat{y}$ such that with probability $\ge 99/100$,

$$\|y - \hat{y}\|_\infty \le (\varepsilon/3)\|y\|_2$$

and therefore have that $\|\hat{y}\|_\infty = \|y\|_\infty \pm (\varepsilon/3)\|y\|_2$. Note that the hash functions for the CountSketch data structure can be stored using $O(r \cdot \log m) = O(\log^2 m)$ bits. If the vector y has entries bounded by $\text{poly}(d)$, then the CountSketch data structure can be stored in $O(t \cdot r \cdot \log d)$ bits. We now note that $\boldsymbol{L}x$ is a C/ε^8 dimensional vector with entries bounded by $\text{poly}(d)$. Hence, using a CountSketch data structure, we can obtain an estimate **Est** that satisfies

$$\mathbf{Est} = \|\boldsymbol{L}x\|_\infty \pm (\varepsilon/3)\|\boldsymbol{L}x\|_2 = \|x\|_\infty \pm \varepsilon\|x\|_2.$$

The two stage sketching procedure can be implemented in a turnstile stream as follows: when we receive an update (i, Δ) to the vector x, we supply the update $(\boldsymbol{h}(i), \boldsymbol{s}(i)\Delta)$ to the CountSketch data structure. The overall space usage of the algorithm is $O(\log d + (\log 1/\varepsilon)^2 + \varepsilon^{-2}\log(1/\varepsilon)\log d)$ bits where we use $O(\log d)$ bits to store the hash functions corresponding to $\boldsymbol{L}$, $O(\log 1/\varepsilon)^2$ bits to store the hash functions corresponding to the CountSketch data structure and $O(\varepsilon^{-2}\log(1/\varepsilon)\log d)$ bits to store the CountSketch table itself.

To process each update in the stream, we require $O(\log 1/\varepsilon)$ time in the Word RAM model with a word size $O(\log d)$ as each update only involves evaluating $O(\log 1/\varepsilon)$ constant wise independent hash functions. □

We will now show that the above is tight up to constant factors.

10.8.1 Space Lower Bound for estimating $\|x\|_\infty$ in a turnstile stream

To lower bound the space complexity of the turnstile streaming algorithm, we reduce from the Augmented Sparse Set-Disjointness problem. We define this communication problem as a combination of the so-called Augmented INDEX problem [CW09] and Sparse Set-Disjointness problem [DKS12]. In this problem, Alice is given sets $A_1, \ldots, A_t \subseteq [n]$ and Bob is given the sets $B_1, \ldots, B_t \subseteq [n]$. Assume that for all $j \in [t]$, $|A_j| = |B_j| = k$. Bob is given an index j and the sets $A_1, \ldots, A_{j-1}$ and has to output using a one-way message M from Alice if $A_j \cap B_j = \emptyset$ or not. Suppose that Alice and Bob have access to a shared random string. We say that a randomized one-way protocol has δ error if for any instance of the Augmented Sparse Set-Disjointness problem, when Alice and Bob run the protocol Π, Bob outputs the correct answer with probability $\geq 1-\delta$. Note that in the one-way protocol, Alice can only send a single message M (possibly randomized using the shared random string) to Bob. The δ-error communication complexity of the Augmented Sparse Set-Disjointness problem is then defined as the minimum over all δ-error protocols of the maximum length, measured in terms of number of bits, of the message sent by Alice over all the inputs. By Yao's minimax principle, we can lower bound the communication complexity of the problem by exhibiting a distribution over the inputs such that any *deterministic* protocol must have a large communication complexity for Bob to output correct answer with probability $\geq 1 - \delta$ (over the distribution of inputs). We now show that there is a small enough constant δ for which the δ-error communication complexity of the Augmented Sparse Set-Disjointness problem is $\Omega(tk \log k)$.

Theorem 10.8.4. *Let t be arbitrary. If $n \geq k^2$, there exists a small enough universal constant δ (independent of t, k and n) such that the δ error randomized communication complexity of the Augmented Sparse Set-Disjointness problem is $\Omega(tk \log k)$.*

Proof. Given parameters k and $\alpha < 1/2$, [DKS12] show that there exists a family $\mathcal{X}$ of $2^{\alpha k \log k}$ subsets of $[k^2]$ such that (i) $|X| = k$ for all $X \in \mathcal{X}$ and (ii) for all $X \neq X' \in \mathcal{X}$, we have $|X \cap X'| \leq \alpha k$. They also show that corresponding to the family $\mathcal{X}$, there is a family $\mathcal{Y}$ of subsets of $[k^2]$ such that (i) $|Y| = k$ for all $Y \in \mathcal{Y}$, (ii) $|\mathcal{Y}| \leq ak \log k$ for an absolute constant $a = a(\alpha)$ (independent of k)

and (iii) for $X \neq X' \in \mathcal{X}$, there is at least one set $Y \in \mathcal{Y}$ such that exactly one of the sets $X \cap Y$ and $X' \cap Y$ is non-empty. The last property implies that the sequence $(\textsc{Disj}(X, Y))_{Y \in \mathcal{Y}}$ is distinct for each $X \in \mathcal{X}$. Here $\textsc{Disj}(X, Y) = 1$ if $X \cap Y = \emptyset$ and 0 otherwise.

We will now define a distribution over the instances of the Augmented Set-Disjointness problem such that any deterministic protocol must have Alice sending a message with $\Omega(tk \log k)$ bits which will prove the theorem by Yao's minimax principle. Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_t \sim \mathcal{X}$ and $\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_t \sim \mathcal{Y}$ be drawn independently. We let $\boldsymbol{X} = (\boldsymbol{X}_1, \dots, \boldsymbol{X}_t)$ denote the sets given to Alice and $\boldsymbol{Y} = (\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_t)$ denote the sets given to Bob. Let $\boldsymbol{i} \sim [t]$ drawn uniformly at random be the index given to Bob. Let $M(\boldsymbol{X})$ be the message sent by Alice when running an arbitrary deterministic protocol with an error δ over the distribution defined by the random variables $\boldsymbol{X}, \boldsymbol{Y}$, and $\boldsymbol{i}$. By the chain rule of entropy,

$$H(\boldsymbol{X} \mid M(\boldsymbol{X})) = \sum_{i \in [t]} H(\boldsymbol{X}_i \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}).$$

As $\boldsymbol{X}_i$ is uniquely identifiable by the sequence $(\textsc{Disj}(\boldsymbol{X}_i, Y))_{Y \in \mathcal{Y}}$, we have

$$\begin{aligned}
H(\boldsymbol{X} \mid M(\boldsymbol{X})) &= \sum_{i \in [t]} H((\textsc{Disj}(\boldsymbol{X}_i, Y))_{Y \in \mathcal{Y}} \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}) \\
&\leq \sum_{i \in [t]} \sum_{Y \in \mathcal{Y}} H(\textsc{Disj}(\boldsymbol{X}_i, Y) \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}) && \text{(sub-additivity)} \\
&= \sum_{i \in [t]} \sum_{y \in \mathcal{Y}} H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i) \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}, \boldsymbol{Y}_i = y) \\
&= \sum_{i \in [t]} |\mathcal{Y}| H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i) \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}, \boldsymbol{Y}_i) && \text{(since } \boldsymbol{Y}_i \text{ is uniform over } \mathcal{Y}) \\
&= |\mathcal{Y}| \sum_{i \in [t]} H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i) \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}, \boldsymbol{Y}).
\end{aligned}$$

Here the last equality follows from the fact that $\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_t$ are mutually independent and are also independent of $\boldsymbol{X}_{<i}$ and $M(\boldsymbol{X})$. As the *deterministic* protocol has δ error over the distribution of the instances we defined above, we have that there is a deterministic function f (which Bob runs to output his answer) such that

$$\mathbf{Pr}_{X,Y,\boldsymbol{i}}[f(M(\boldsymbol{X}), \boldsymbol{i}, \boldsymbol{Y}, \boldsymbol{X}_{<\boldsymbol{i}}) = \textsc{Disj}(\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}})] \geq 1 - \delta.$$

We then have that with probability $1 - \sqrt{\delta}$ over $\boldsymbol{i}$ that

$$\mathbf{Pr}_{X,Y}[f(M(\boldsymbol{X}), \boldsymbol{i}, \boldsymbol{Y}, \boldsymbol{X}_{<\boldsymbol{i}}) = \textsc{Disj}(\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}})] \geq 1 - \sqrt{\delta}.$$

Let $\textsc{Good} \subseteq [t]$ denote all the indices i for which the above holds. We have $|\textsc{Good}| \geq (1 - \sqrt{\delta})t$. We now note that $\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i)$ is a 0/1 random variable. By Fano's inequality, for all $i \in \textsc{Good}$, we obtain

$$H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i) \mid M(\boldsymbol{X}), \boldsymbol{Y}, \boldsymbol{X}_{<i}) \le H(\sqrt{\delta}).$$

For $i \notin \textsc{Good}$, we simply have $H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i) \mid M(\boldsymbol{X}), \boldsymbol{Y}, \boldsymbol{X}_{<i}) \le H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i)) \le 1$. Thus,

$$\begin{aligned}
H(\boldsymbol{X} \mid M(\boldsymbol{X})) &\le |\mathcal{Y}| \sum_{i \in [t]} H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i) \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}, \boldsymbol{Y}) \\
&= |\mathcal{Y}| \sum_{i \in \textsc{Good}} H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i) \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}, \boldsymbol{Y}) \\
&\qquad + |\mathcal{Y}| \sum_{i \notin \textsc{Good}} H(\textsc{Disj}(\boldsymbol{X}_i, \boldsymbol{Y}_i) \mid M(\boldsymbol{X}), \boldsymbol{X}_{<i}, \boldsymbol{Y}) \\
&\le |\mathcal{Y}| t(1 - \sqrt{\delta}) H(\sqrt{\delta}) + |\mathcal{Y}| t \sqrt{\delta} \\
&= |\mathcal{Y}| t(\sqrt{\delta} + (1 - \sqrt{\delta}) H(\sqrt{\delta})).
\end{aligned}$$

Now, $H(\boldsymbol{X} \mid M(\boldsymbol{X})) \ge H(\boldsymbol{X}) - H(M(\boldsymbol{X}))$ by the chain rule and sub-additivity which implies from the above inequality that

$$H(M(\boldsymbol{X})) \ge H(\boldsymbol{X}) - |\mathcal{Y}| t(\sqrt{\delta} + (1 - \sqrt{\delta}) H(\sqrt{\delta})).$$

As $H(\boldsymbol{X}) = H((\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t)) = tH(\boldsymbol{X}_1) = t\alpha k \log k$, we have

$$\begin{aligned}
H(M(\boldsymbol{X})) &\ge t\alpha k \log k - tak \log k(\sqrt{\delta} + (1 - \sqrt{\delta}) H(\sqrt{\delta})) \\
&\ge tk \log k(\alpha - a(\sqrt{\delta} + (1 - \sqrt{\delta}) H(\sqrt{\delta}))).
\end{aligned}$$

Now we note that $H(\sqrt{\delta}) \le 2\delta^{1/4}$ and get $H(M(\boldsymbol{X})) \ge tk \log k(\alpha - a(\sqrt{\delta} + 2\delta^{1/4}))$. As $a = a(\alpha)$ is purely a function of α independent of k, by picking δ to be a small enough function of α, we get $H(M(\boldsymbol{X})) \ge (\alpha/2) tk \log k$. Finally, this implies that $\max_{\boldsymbol{X}} |M(\boldsymbol{X})| \ge H(M(\boldsymbol{X})) \ge (\alpha tk \log k)/2$. Picking $\alpha = 1/2$, we obtain that there is a small enough constant δ and a product distribution $\mathcal{D}$ over $\mathcal{X} \otimes \mathcal{Y} \otimes [t]$ such that any *deterministic* one-way protocol that solves the Augmented Sparse Set-Disjointness problem with probability $\ge 1 - \delta$ over the distribution $\mathcal{D}$ must have a communication complexity of $\Omega(tk \log k)$ bits. $\square$

Using the above lower bound, we can now show that any turnstile streaming algorithm that approximates the ℓ_∞ norm of a d dimensional vector x with integer coordinates bounded in absolute value by $\mathrm{poly}(d)$ up to an additive error of $\varepsilon\|x\|_2$ must use $O(\varepsilon^{-2} \log(1/\varepsilon) \log d)$ bits of space.

Theorem 10.8.5. *There exists a small enough constant δ such that if $\varepsilon \ge 6((\log d)/d)^{1/4}$, any turnstile streaming algorithm that estimates $\|x\|_\infty$ up to an additive error of $\varepsilon\|x\|_2$ with probability $\ge 1 - \delta$, of a d-dimensional vector x with integer entries bounded in absolute value by $\mathrm{poly}(d)$, must use a space of*

$\Omega(\varepsilon^{-2}\log(1/\varepsilon)\log(d))$ *bits.*

Proof. Let $t = \log d$, $k = 1/\varepsilon^2$ and $n = 1/\varepsilon^4$. From the above theorem, the Augmented Sparse Set-Disjointness problem with these parameters has a randomized communication complexity of $\Omega(\varepsilon^{-2}\log(1/\varepsilon)\log d)$ bits.

Suppose given an instance of the Augmented Sparse Set-Disjointness problem, Alice computes a vector $x \in \mathbb{R}^{(\log d)\cdot 1/\varepsilon^4}$ as follows: the vector x is divided into $\log d$ blocks—one for each of the sets $A_1,\ldots,A_{\log d}$. The i-th block of vector x, for $i = 1,\ldots,t$, is defined to be $10^{t-i}\cdot a_i$ where a_i is a binary-vector representation of the set A_i. As $t = \log d$, we obtain that $\|x\|_\infty \le \operatorname{poly}(d)$. Let $\boldsymbol{M}$ be a randomized turnstile streaming algorithm for estimating $\|x\|_\infty$. Let $\boldsymbol{M}(x)$ be the state of the turnstile streaming algorithm after feeding the coordinates of the vector x to $\boldsymbol{M}$. Alice transmits the state $\boldsymbol{M}(x)$ to Bob. As Bob has an index i and the sets $A_1,\ldots,A_{i-1}$, Bob can construct a vector $y \in \mathbb{R}^{(\log d)\cdot 1/\varepsilon^4}$ such that the j-th block of vector y for $j = 1,\ldots,i-1$ is the same as the j-th block of the vector x. The rest of the blocks of y are set to be 0. We now note that the only non-zero blocks of the vector $x - y$ are $i, i+1,\ldots,t$.

Bob feeds the updates corresponding to the vector $-y$ to the streaming algorithm $\boldsymbol{M}$ starting with the state $\boldsymbol{M}(x)$ to obtain $\boldsymbol{M}(x-y)$. Finally, Bob defines a vector z with the i-th block being the vector $10^{t-i}\cdot b_i$ where b_i is the binary vector corresponding to the set B_i. The rest of the blocks of z are set to be 0. Bob finally updates the state of the streaming algorithm to obtain $\boldsymbol{M}(x-y+z)$. If $A_i \cap B_i = \emptyset$, we have $\|x-y+z\|_\infty = 10^{t-i}$ and if $A_i \cap B_i \ne \emptyset$, then $\|x-y+z\|_\infty = 2\cdot 10^{t-i}$. Additionally, we have $\|x-y+z\|_2^2 \le 4\cdot 10^{2(t-i)}\cdot k + \sum_{j=i+1}^{t} 10^{2(t-j)}\cdot k \le 5\cdot 10^{2(t-i)}\cdot k$. Thus, $\|x-y+z\|_2 \le 3\cdot 10^{t-i}\cdot(1/\varepsilon)$ since $k = 1/\varepsilon^2$. Thus, an approximation of $\|x-y+z\|_\infty$ up to an additive error of $(\varepsilon/6)\|x-y+z\|_2$ lets Bob output the correct answer for the instance of Augmented Sparse Set-Disjointness problem.

By the lower bound on communication complexity of the Augmented Sparse Set-Disjointess problem, we obtain that any turnstile streaming algorithm that outputs, with probability $1-\delta$ for a small enough constant δ, an approximation to $\|x\|_\infty$ up to an additive error of $(\varepsilon/6)\|x\|_\infty$, for a $(1/\varepsilon^4)\log d$ dimensional vector x with coordinates of absolute values bounded by $\operatorname{poly}(d)$, must use $\Omega(\varepsilon^{-2}\log(1/\varepsilon)\log d)$ bits. As $\varepsilon \ge ((\log d)/d)^{1/4}$ implies $d \ge \varepsilon^{-4}\log d$, we obtain the result. □

Note that the above lower bound crucially uses that the algorithm is a turnstile streaming algorithm and does not hence lower bound the space complexity of the algorithms in the insertion-only streams where only nonnegative updates are allowed to the vector x being maintained in the stream.

10.8.2 Tighter bounds for vectors with large $\|x\|_\infty$

The lower bound in the previous section shows that $\Theta(\varepsilon^{-2}\log(d)\log(1/\varepsilon))$ bits of space is both necessary and sufficient to approximate $\|x\|_\infty$ up to an additive error $\varepsilon\|x\|_2$. The hard instance in the lower bound has the property that $\|x\|_\infty = O(\varepsilon\|x\|_2)$. We show that assuming the vector x

satisfies, $\|x\|_\infty \ge c\|x\|_2$ for a constant c, we can beat the lower bound and obtain a better than $\varepsilon\|x\|_2$ additive error using $O(\varepsilon^{-2}\log d)$ bits of space. Note that the condition $\|x\|_\infty \ge c\|x\|_2$ is natural in certain settings where the coordinates of the vector x follow the Zipf's law. We prove the following theorem.

Theorem 10.8.6. *Suppose a d dimensional vector x is being maintained in a turnstile stream. Assume that the coordinates of x are integers and are bounded in absolute value by $\mathrm{poly}(d)$. If x is such that $\|x\|_\infty \ge c\|x\|_2$ for a universal constant c, then there is an algorithm that uses $O(\varepsilon^{-2}\log d)$ bits of space and outputs an estimate **Est** that satisfies*

$$\boldsymbol{Est} = \|x\|_\infty \pm \varepsilon\|x\|_2$$

with probability $\ge 9/10$. The update time of the algorithm is $O(\log 1/\varepsilon)$ in the Word RAM model with a word size $\Omega(\log d)$.

Proof. Without loss of generality, assume $\varepsilon \le c/10$. Let $\textsc{Large} = \{i \mid |x_i| \ge (c/2)\|x\|_2\}$. We have $|\textsc{Large}| \le 4/c^2$. If $\boldsymbol{L} : \mathbb{R}^d \to \mathbb{R}^{d'}$ is a randomized linear map to $d' = \mathrm{poly}(1/\varepsilon)$ dimension constructed using a 2-wise independent hash function $\boldsymbol{h}$ and a 4-wise independent sign function $\boldsymbol{s}$ as in Lemma 10.8.1, then we have that

1. all the coordinates in $\textsc{Large}$ are hashed to different coordinates,
2. $\|\boldsymbol{L}x\|_2^2 \le (1+\varepsilon^8)\|x\|_2^2$,
3. for all $i \notin \textsc{Large}$, $|(\boldsymbol{L}x)_{\boldsymbol{h}(i)}| \le (c/2+\varepsilon^2/8)\|x\|_2 \le (3c/5)\|x\|_2$, and
4. for all $i \in \textsc{Large}$, $|(\boldsymbol{L}x)_{\boldsymbol{h}(i)}| = |x_i| \pm (\varepsilon^2/8)\|x\|_2$.

Now, let $r = O(\log 1/\varepsilon)$, $t = O(1/\varepsilon^2)$ and $b = 1/\varepsilon$. Instantiate a CountSketch data structure $\mathsf{CS} : \mathbb{R}^{d'} \to \mathbb{R}^{rt}$ with these parameters and derandomized using HashPRG as in Theorem 10.6.2 with a word size $w = \Omega(\log d)$. From Theorem 10.6.2, the parameters (random seed for HashPRG) of the map CS and the value $\mathsf{CS}(\boldsymbol{L}x)$ can be stored using $O(rt + b\log_b d') = O(\varepsilon^{-2}\log 1/\varepsilon + \varepsilon^{-1}\log_{\varepsilon^{-1}}\mathrm{poly}(1/\varepsilon)) = O(\varepsilon^{-2}\log 1/\varepsilon)$ *words* of space. We also have that the update time of the CountSketch data structure instantiated with these parameters is $O(\log 1/\varepsilon)$ in the Word RAM model with a word size $\Omega(\log d)$. If x receives a turnstile update (i, Δ), then

$$\boldsymbol{L}(x+\Delta e_i) = \boldsymbol{L}x + \Delta\cdot(\boldsymbol{L}e_i).$$

By definition of the map $\boldsymbol{L}$, the vector $\boldsymbol{L}e_i$ is nonzero in exactly one coordinate $\boldsymbol{h}(i)$. Thus, we further obtain

$$\mathsf{CS}(\boldsymbol{L}(x+\Delta e_i)) = \mathsf{CS}(\boldsymbol{L}x + \boldsymbol{s}(i)\Delta e_{\boldsymbol{h}(i)}).$$

Now, by Theorem 10.6.2, the vector $\mathsf{CS}(\boldsymbol{L}(x+\Delta e_i))$ can be computed using the value of $\mathsf{CS}(\boldsymbol{L}x)$ in time $O(r\log_b d') = O(\log 1/\varepsilon)$ time in Word RAM model. Thus, the randomized two-level sketch $\mathsf{CS}\circ\boldsymbol{L}$

can be applied to the underlying vector x in a turnstile stream using a total space of $O(\varepsilon^{-2}\log 1/\varepsilon)$ *words* of space and each turnstile update can be processed in $O(\log 1/\varepsilon)$ time on a Word RAM machine with a word size $\Omega(\log d)$.

Theorem 10.6.2 also gives the following recovery guarantees: for any $i \in [d']$ and $\alpha < 1$, we can recover a value $\widehat{(\boldsymbol{L}x)}_i$ such that

$$\mathbf{Pr}_{\mathsf{CS}}[|\widehat{(\boldsymbol{L}x)}_i - (\boldsymbol{L}x)_i| \ge \alpha\varepsilon\|\boldsymbol{L}x\|_2] \le \exp(-\alpha^2 r) + 1/\text{poly}(d).$$

Setting $\alpha = 1/\sqrt{\log 1/\varepsilon}$, a union bound over the indices in the set $h(\text{LARGE}) \subseteq [d']$ gives that with probability $\ge 99/100$ over CS, for all $i \in h(\text{LARGE})$, $|\widehat{(\boldsymbol{L}x)}_i - (\boldsymbol{L}x)_i| \le (\varepsilon/\sqrt{\log 1/\varepsilon})\|\boldsymbol{L}x\|_2$ and setting $\alpha = 1$ and a union bound over all the coordinates $i \in [d']$ gives that with probability $\ge 99/100$, for all $i \in [d]$, $|\widehat{(\boldsymbol{L}x)}_i - (\boldsymbol{L}x)_i| \le \varepsilon\|\boldsymbol{L}x\|_2$. Conditioned on the properties of the map $\boldsymbol{L}$ above, overall, we obtain that with a probability $\ge 9/10$,

$$\|\boldsymbol{L}x\|_\infty = \|x\|_\infty \pm \frac{2\varepsilon}{\sqrt{\log 1/\varepsilon}}\|x\|_2.$$

Hence, $\|x\|_\infty$ can be estimated to an additive error of $(\varepsilon/\sqrt{\log 1/\varepsilon})\|x\|_2$ using only $O(\varepsilon^{-2}\log 1/\varepsilon)$ words of space and the time to update the state in a turnstile stream is $O(\log 1/\varepsilon)$ in the Word RAM model with a word size $\Omega(\log d)$. Setting $\varepsilon' = 2\varepsilon/\sqrt{\log 1/\varepsilon}$, we obtain the result. □

10.9 Conclusions an Open Questions

In this chapter, we construct a new pseudorandom generator that has a space-vs-time trade-off that lets us obtain space-optimal streaming algorithms with a fast update time for a number of problems. Our key insight is that for a number of applications, we do not require that the pseudorandom generator *fool* the full streaming algorithms. By carefully defining *analysis algorithms* that capture the necessary properties of the instantiated random variables and use a much smaller amount of space compared to the full algorithm, we can show that a much weaker PRG suffices to obtain the required properties. This technique allows us to use PRGs that have a fast retrieval time.

Our algorithm for F_p approximation for $p \in (0, 2)$ requires that $\varepsilon < 1/d^c$ for a small constant c and has an update time of $O(\log d)$. The earlier algorithm of [KNPW11] has an update time of $O(\log^2(1/\varepsilon)\log\log(1/\varepsilon))$. It would be interesting to obtain space-optimal algorithms with an update time $O(\log d)$ for all $\varepsilon \ge 1/\text{poly}(d)$.

Chapter 11

High-Dimensional Geometric Streaming for Nearly Low Rank Data

11.1 Introduction

Modern datasets are usually very high dimensional and have numerous data points. Storing the entire dataset to analyze them is often impractical and in certain settings impossible. In recent years, streaming algorithms have emerged as a way to process and understand the datasets in both a space and time efficient manner. In a single-pass streaming setting, an algorithm is allowed to make only a single pass over the entire dataset and is required to output a "summary" of the dataset that is useful to solve a certain problem. We focus on streaming algorithms for high-dimensional geometric problems such as subspace approximation, width estimation, etc. Suppose we are given a set of d-dimensional points $a_1, \ldots, a_n$ and an integer parameter $k \le d$. Given a subspace V, we define $d(a, V)$ to be distance between the point a and subspace V given by $\min_{v \in V} \|a - v\|_2$. The ℓ_p subspace approximation problem [DTV11], for $p \in [1, \infty]$, asks to find a k-dimensional subspace that minimizes $(\sum_{i=1}^{n} d(a_i, V)^p)^{1/p}$.

Note that for $p = \infty$, we want to find a k-dimensional subspace that minimizes the maximum distance from the given set of points. Related to the ℓ_∞ subspace approximation problem is the widely studied outer $(d-k)$ radius estimation problem [VVYZ07] which instead asks for a k-dimensional flat[1] F that minimizes $\max_{i \in [n]} d(a_i, F)$. The outer $(d-k)$ radius is a measure of how far the point set is from being inside a k-dimensional flat. [VVYZ07] give a polynomial time algorithm for approximating the outer $(d-k)$ radius up to an $O(\sqrt{\log n})$ multiplicative factor. Their algorithm is based on rounding of a semidefinite program (SDP) relaxation. When n is very large, their algorithm is not practical and cannot be implemented in the streaming setting. We give a time and space efficient single pass streaming algorithm that approximates the outer $(d-k)$ radius up to a $\widetilde{O}(\sqrt{k}\log(n\kappa))$ factor, where κ is a suitably defined condition number. Typically, the value of k used is much smaller

[1] A k dimensional flat is defined as a k dimensional subspace that is translated by some c.

than n and d since in many settings, we have that the $n \times d$ matrix A is a noisy version of an underlying rank k matrix for a small value of k.

Our main contribution is a simple ***deterministic*** algorithm the constructs a *strong coreset* for approximating $\max_i d(a_i, V)$ for any k-dimensional subspace V in a single-pass streaming setting. When run on the stream of points $a_1, \ldots, a_n$, our algorithm selects a subset $S \subseteq [n]$ of points, $|S| = O(k \log^2(n\kappa))$ such that for all k dimensional subspaces V, $\max_{i\in S} d(a_i, V) \le \max_{i\in[n]} d(a_i, V) \le O(\sqrt{k}\log(n\kappa)) \max_{i\in S} d(a_i, V)$. We stress that our coreset can be used to approximate the max distance of the point set to *any* k-dimensional subspace and hence it is termed a strong coreset. We prove:

Theorem 11.1.1 (Informal). *Given a parameter k and n points $a_1, \ldots, a_n \in \mathbb{R}^d$, Algorithm 11.1 selects a subset $S \subseteq [n]$ of points, $|S| = O(k \log^2 n\kappa)$ such that for all k-dimensional subspaces V,*

$$\max_{i\in S} d(a_i, V) \le \max_{i\in[n]} d(a_i, V) \le O(\sqrt{k}\log n\kappa) \max_{i\in S} d(a_i, V).$$

The streaming algorithm requires only enough space to store $O(k \log^2 n\kappa)$ rows of A and can be implemented in time $O(\text{nnz}(A)\log n + d\,\text{poly}(k, \log n\kappa))$ if one is allowed randomization.

In this result and its applications throughout this chapter, the condition number κ can be replaced with $n^{O(k)}$ assuming that all the entries in the points are integers bounded in absolute value by $\text{poly}(n)$. Although under suitable assumptions about the "noise" in the process generating the data, κ will be much lower.

We then show using a simple reduction that the above theorem can be used to approximate the outer $(d-k)$ radius by running the streaming algorithm on the point set $a_2 - a_1, \ldots, a_n - a_1$.

We then turn to the ℓ_p subspace approximation for general $p \in [1, \infty)$. We observe that an instance of the ℓ_p subspace approximation problem can be turned to an ℓ_∞ subspace approximation by using the so-called min-stability property of exponential random variables. We scale each input point with appropriately chosen independent random variables and feed the scaled points to Algorithm 11.1. We obtain the following result:

Theorem 11.1.2 (Informal). *Given $p \ge 1$, a dimension parameter k, and n points $a_1, \ldots, a_n \in \mathbb{R}^d$, there is a randomized streaming algorithm that selects a subset $S \subseteq [n]$, $|S| = O(k \log^2 n\kappa)$ and assigns a weight $w_i \ge 0$ for $i \in S$ such that if*

$$\widetilde{V} = \arg\min_{k\text{-dim } V} \max_{i\in S} w_i \cdot d(a_i, V),$$

then

$$\frac{(\sum_{i=1}^n d(a_i, \widetilde{V})^p)^{1/p}}{\min_{k\text{-dim } V}(\sum_{i=1}^n d(a_i, V)^p)^{1/p}} \le k^{1/2+2/p}\,\text{poly}(\log^{1+3/p} n\kappa).$$

The algorithm only uses $O(d \cdot k \log^2 n\kappa)$ bits of space and runs in $O(\text{nnz}(A)\log n + d\,\text{poly}(k, \log n))$ time.

While exponential random variables have been previously used in the context of ℓ_p subspace embeddings and ℓ_p moment estimation in streams, as far as we are aware, ours is the first work to use them in the context of subspace approximation.

We then show that recent algorithms of [WY22] can be improved using our coreset construction algorithm when the data points $a_1, \ldots, a_n$ are approximately spanned by a low rank subspace. They give streaming algorithms for a host of geometric problems such as width estimation, volume estimation, Löwner-John ellipsoid, etc. The main ingredient of their algorithms is a deterministic ℓ_∞ subspace embedding: their algorithm streams through rows of an $n \times d$ matrix A and selects a subset of rows $S \subseteq [n]$, $|S| = O(d \log n)$ with the property that for all x,

$$\|A_S x\|_\infty \le \|Ax\|_\infty \le \sqrt{d \log n}\|A_S x\|_\infty.$$

Here $\|x\|_\infty \coloneqq \max_i |x_i|$ and A_S is the matrix A restricted to only those rows in S. When the matrix A has rank d, their algorithm necessarily needs $\Omega(d^2)$ bits of space which is prohibitive when d is very large. In practice, many matrices A are very well approximated by a matrix with far lower rank than d even when the rank of the matrix A is d. Suppose A is well approximated by a rank k matrix in the sense that there is a k-dimensional subspace V such that all the rows of A are not very far from V. We show that if S is the coreset constructed by Algorithm 11.1, then for all *unit vectors* x, $\|A_S x\|_\infty \le \|Ax\|_\infty \le (C\sqrt{k}\log n\kappa)\|A_S x\|_\infty + C\Delta \log n\kappa$, where Δ denotes the optimal rank-k ℓ_∞ subspace approximation cost of the matrix A. Thus, $\|A_S x\|_\infty$ can be used to approximate $\|Ax\|_\infty$ well when Δ is small.

11.1.1 Previous Work

The rank-k ℓ_∞ subspace approximation problem and more generally the rank-k ℓ_∞ flat approximation problem have been previously studied for different values of k. As discussed earlier, [VVYZ07] give an SDP-based algorithm that can compute an $O(\sqrt{\log n})$ factor approximation for all values of k. Being SDP-based, the algorithm is impractical in the streaming setting and when the number of points n is very large. We shall mostly discuss previous works relevant in the streaming setting.

For specific values of $k = 0$ and $k = d - 1$, Agarwal and Sharathkumar [AS15] study upper and lower bounds on streaming algorithms. For $k = 0$, also known as the minimum enclosing ball (MEB) problem, they give a streaming algorithm that is a $(1 + \sqrt{3})/2$ approximation and show that there is a small enough constant α such that any α approximation algorithm must use $\min(n, \exp(d^{1/3}))$ space thereby showing that there are no small-space streaming algorithms with a better than α approximation. For $k = d - 1$, the so-called width estimation problem, they showed that any algorithm that approximates the cost up to a multiplicative $\Theta(d^{1/3})$ factor must use $\Omega(n, \exp(d^{1/3}))$ bits of space again ruling out small-space algorithms with better than $d^{1/3}$ approximation factor.

Later, Chan and Pathak [CP14] improved the approximation ratio of the algorithm of [AS15] to $(1 + \sqrt{2})/2$ for the MEB problem.

Recently, [TWZ+22] give an algorithm to construct a coreset for the ℓ_∞ subspace approximation problem with a size $\widetilde{O}(k^{3k})$. While an offline coreset construction can be converted into a streaming coreset construction using the merge-and-reduce procedure, the exponential dependence in k makes their algorithm impractical as compared to over algorithm which needs to store $O(k\log(n\kappa)^2)$ input points.

For the ℓ_1 subspace approximation problem, [FMSW10] give a streaming algorithm to construct a coreset with $\widetilde{O}\left(d\left(\frac{k\cdot 2^{O(\sqrt{\log n})}}{\varepsilon^2}\right)^{\mathrm{poly}(k)}\right)$ points that can be used to compute a $1+\varepsilon$ approximation. When n and d are large, the space requirement of the coreset is infeasible. In comparison, although our algorithms do not give $1+\varepsilon$ approximation, we can compute $\mathrm{poly}(k, \log n\kappa)$ approximations using only space necessary to store $\mathrm{poly}(k, \log n\kappa)$ points which is much smaller than the coreset constructed by their algorithm.

For all values of p, Kerber and Raghavendra [KR14] give a dimensionality reduction procedure by showing that projecting the points to a random $O(k^2(\log k/\varepsilon \cdot \log n)/\varepsilon^3)$-dimensional space preserves the ℓ_p subspace approximation[2] cost. For $p = \infty$, their algorithm combined with the coreset construction algorithm of Woodruff and Yasuda [WY22] can be used to approximate the ℓ_∞ subspace approximation up to $\mathrm{poly}(k, \log n)$ factors. But since the d-dimensional "information" is destroyed by the projection, we cannot recover a solution in the d-dimensional space. In comparison, for $p = \infty$, we give a practical algorithm to construct a strong coreset that lets us approximate the maximum distance to any k dimensional subspace and for general p, we give a polynomial time algorithm that can output a "d-dimensional" approximate solution.

For $p \notin \{1, 2, \infty\}$, much less is known in the streaming setting. In the offline setting, Deshpande and Varadarajan [DV07] gave a sampling based algorithm for all $p \ge 1$ that outputs a bicriteria solution for the ℓ_p subspace approximation problem. Later [DTV11] gave a polynomial time $O(\sqrt{p})$ factor approximation algorithm for the ℓ_p subspace approximation problem for all $p \ge 2$. Assuming the Unique Games Conjecture, they show that it is hard to approximate the cost to a smaller than $O(\sqrt{p})$ factor. For $1 \le p \le 2$, [CW15] gave an input sparsity time algorithm that computes a $1+\varepsilon$ approximation but they have an $\exp(\mathrm{poly}(k/\varepsilon))$ term in their running time. The $O(\sqrt{p})$ factor approximation algorithm of [DTV11] is based on convex relaxations and is not applicable in the streaming setting. In a recent work, Deshpande and Pratap [DP23] observed the lack of streaming algorithms for ℓ_p subspace approximation that also have the subset selection property that our coresets have. They give a subset selection algorithm for the ℓ_p subspace approximation problem but their results have a weaker additive error guarantee. They leave open the subset selection algorithms that give a multiplicative approximation to the ℓ_p subspace approximation problem. In a recent work, Woodruff and Yasuda [WY23] answered the question of [DP23] in the affirmative by giving a subset selection algorithm the computes a strong coreset with $O((k/\varepsilon)^{O(p)}\,\mathrm{polylog}(n))$ rows that can approximate the cost of any k-dimensional space up to a $1 \pm \varepsilon$ factor. Selecting $k^{O(p)}$ rows is prohibitive when p is large. Our work makes progress on this question by removing the exponential dependence in p

[2]They prove their result for the more general problem of subspace clustering

although at the cost of only being able to compute a multiplicative $\text{poly}(k, \log n\kappa)$ approximation to the problem.

Relevance to Machine Learning. Our work continues the long line of work in the area of subspace approximation and low rank approximation with different error metrics that has been of interest in the Machine Learning community. Previous works study problems such as ℓ_1 subspace approximation [HM13], entry wise ℓ_p low rank approximation [CGK+17, DWZ+19], Column subset selection for entrywise ℓ_p norm and other error metrics [SWZ19]. Our algorithms for geometric streaming problems such as convex hull estimation have applications for robust classification [PF01, FNM07].

11.2 Preliminaries

If $S \subseteq [n]$, then A_S denotes the submatrix formed by the rows in the set S. Given indices $i < j$, we use $A_{i:j}$ to denote the matrix formed by the rows $a_i, \ldots, a_j$.

For an arbitrary k dimensional subspace $V \in \mathbb{R}^d$, we use $\mathbb{P}_V$ to denote the orthogonal projection matrix onto the subspace V, i.e., for any $x \in \mathbb{R}^d$, $\mathbb{P}_V \cdot x$ is the closest (in Euclidean norm) vector to x in V. So, $d(x, V) = \|(I - \mathbb{P}_V)x\|_2$ and $\|A(I - \mathbb{P}_V)\|_{\infty,2} = \max_i \|(I - \mathbb{P}_V)a_i\|_2 = \max_i d(a_i, V)$.

11.3 ℓ_∞ Low Rank Approximation and Outer Radius

As discussed in the introduction, given a matrix A with rows $a_1, \ldots, a_n$ that arrive in a stream, we want to compute a *strong coreset*, i.e., a subset $S \subseteq [n]$ such that for all k-dimensional subspaces V,

$$1 \leq \frac{\max_{i \in [n]} d(a_i, V)}{\max_{i \in S} d(a_i, V)} \leq f$$

for a small *distortion* f. Consider the following simple algorithm: we initiate $S \leftarrow \emptyset$ and stream through the rows $a_1, \ldots, a_n$. When processing the row a_i, if there exists a k-dimensional subspace V such that $d(a_i, V)^2 > \sum_{i \in S} d(a_i, V)^2$, we update $S \leftarrow S \cup \{ i \}$. Otherwise, we proceed to the next row without updating S. Consider the set S at the end of the stream and let V be an arbitrary k dimensional subspace. We shall now argue that A_S is a strong coreset with a distortion at most $\sqrt{|S|}$.

Let V be an arbitrary k-dimensional subspace of $\mathbb{R}^d$. Let $i^* = \arg\max_i d(a_i, V)$ be the index of the row *farthest* from V. Consider the following cases: if $i^* \in S$, then we have $\max_{i \in [n]} d(a_i, V) = d(a_{i^*}, V) = \max_{i \in S} d(a_i, V)$ and therefore A_S has *no distortion* for V. In case the index $i^* \notin S$, then

$d(a_{i^*}, V)^2 \leq \sum_{i \in S, i < i^*} d(a_i, V)^2$ since otherwise we would have added i^* to S. Thus,

$$\max_i d(a_i, V) = d(a_{i^*}, V) \leq \sqrt{\sum_{i \in S} d(a_i, V)^2} \qquad (11.1)$$
$$\leq \sqrt{|S|} \max_{i \in S} d(a_i, V)$$

and therefore A_S is a strong coreset with a distortion at most $\sqrt{|S|}$. Now, if we can show that S can not be too large, we obtain that A_S is a strong coreset with a small distortion.

To show that S is not too large, we appeal to rank-k *online* ridge leverage scores, a generalization of the so-called *ridge leverage scores*. In the offline setting, ridge leverage scores have been employed by Cohen, Musco, and Musco [CMM17] as a suitable modification of the usual ℓ_2-leverage scores to obtain fast algorithms for ℓ_2 low rank approximation. Later, [BDM+20] defined online ridge leverage scores and showed that they can be used to compute low rank approximations in the *online* model. They also showed that for well-conditioned instances, the sum of the online ridge leverage scores is small. Our main observation is that for the set S constructed as described, the online rank-k ridge leverage score of *every* row in A_S is large. As the sum of online rank-k ridge leverage scores is not large, which we prove, we obtain that there cannot be too many rows in A_S.

One issue we have to solve to implement this algorithm is given a_i and the set S after processing $a_1, \ldots, a_{i-1}$, how can we efficiently know if there exists a rank-k subspace V such that $d(a_i, V)^2 > \sum_{i \in S} d(a_i, V)^2$? Online ridge leverage scores again come to rescue. We show that if we modify the above described algorithm to instead add i to S when its "online rank-k ridge leverage score" is large with respect to A_S, then the set S computed at the end of the process is again a strong coreset with a distortion of at most $\sqrt{|S|}$.

11.3.1 Online Rank-k Ridge Leverage Scores

Let A be an arbitrary matrix with rows $a_1, \ldots, a_n \in \mathbb{R}^d$ and let $k \leq d$ be a rank parameter. Let $\lambda_i = \frac{\|A_{1:i} - [A_{1:i}]_k\|_F^2}{k}$ be the i-th ridge parameter. Note that $\lambda_i = 0$ if and only if $\operatorname{rank}(A_{1:i}) \leq k$. We define the "rank-$k$ online ridge leverage score" of the row a_{i+1} to be

$$\tau_{i+1}^{\mathrm{OL},k}(A) \coloneqq \begin{cases} 1 \text{ if } \lambda_i = 0 \text{ and } a_{i+1} \notin \operatorname{rowspace}(A_{1:i}) \\ \min(1, a_{i+1}^{\mathrm{T}}(A_{1:i}^{\mathrm{T}} A_{1:i} + \lambda_i \cdot I)^+ a_{i+1}) \text{ o.w.} \end{cases}$$

The online rank-k ridge leverage scores help us capture the "rank-k information" of the matrix A as the rows are revealed.

Algorithm 11.1: Minimize Distance to a Subspace

```
Input: A matrix $A$ as a stream of rows $a_1, \ldots, a_n \in \mathbb{R}^d$, a rank parameter $k$
Output: A subset $S \subseteq [n]$
1 $S \leftarrow \emptyset, \lambda \leftarrow 0$                                  // Algorithm stores $A_S$
2 for $t = 1, \ldots, n$ do
3     if $\lambda = 0$ and $a_t \notin rowspace(A_S)$ then
4         $S \leftarrow S \cup \{t\}$
5     else if $a_t^{\mathrm{T}}(A_S^{\mathrm{T}} A_S + \lambda \cdot I)^{+} a_t \geq 1/(1 + 1/k)$ then
6         $S \leftarrow S \cup \{t\}$
7     $\lambda \leftarrow \|A_S - [A_S]_k\|_{\mathrm{F}}^2 / k$                  // $\lambda$ changes only when $S$ changes
8 end
9 return $S$
```

11.3.2 An Efficient Algorithm

Our full coreset construction algorithm is described in Algorithm 11.1. In the algorithm, we select a subset of rows S online in the following way: a new row a_t is added to the set S if the rank-k online ridge leverage score of the row a_t with respect to the matrix $A_{S \cup t}$ is at least $1/(1 + 1/k)$.

We will first show that the set S computed by the algorithm defines a matrix A_S that is a strong coreset with a distortion at most $\sqrt{|S|}$. Let $S_t \coloneqq S \cap [t]$ be the subset of rows that have been selected by the algorithm after processing $a_1, \ldots, a_t$ and let a_{t+1} is the row being processed. We prove the following lemma:

Lemma 11.3.1. *Let t be arbitrary and let $S_t \coloneqq S \cap [t]$ be the subset of rows selected by Algorithm 11.1 after processing the rows $a_1, \ldots, a_t$. If there exists a rank k subspace V such that*

$$d(a_{t+1}, V)^2 \geq \sum_{i \in S_t} d(a_i, V)^2,$$

then the algorithm adds the row $t + 1$ to the set S that it maintains.

Proof. Assume that there is a k-dimensional subspace V such that $d(a_{t+1}, V)^2 > \sum_{i \in S_t} d(a_i, V)^2$ where $S_t = S \cap [t]$ is the set of rows selected by the algorithm after processing the rows $a_1, \ldots, a_t$.

If $\sum_{i \in S_t} d(a_i, V)^2 = 0$, then $\mathrm{rank}(A_{S_t}) \leq k$ and $\mathrm{rowspace}(A_{S_t}) \subseteq V$. Since $d(a_{t+1}, V) > 0$, we have $a_{t+1} \notin V$ which implies $a_{t+1} \notin \mathrm{rowspace}(A_{S_t})$ and therefore the algorithm adds $t + 1$ to the set S.

Now, suppose $\sum_{i \in S_t} d(a_i, V)^2 > 0$. Let $\mathbb{P}_V$ be the orthogonal projection matrix onto the subspace V and define

$$x^* \coloneqq \frac{(I - \mathbb{P}_V) a_{t+1}}{\|(I - \mathbb{P}_V) a_{t+1}\|_2}.$$

Using the fact that $(I - \mathbb{P}_V)$ is also a projection matrix, we obtain

$$|\langle a_{t+1}, x^* \rangle|^2 = \frac{(a_{t+1}^{\mathrm{T}}(I - \mathbb{P}_V)a_{t+1})^2}{\|(I - \mathbb{P}_V)a_{t+1}\|_2^2} = \frac{\|(I - \mathbb{P}_V)a_{t+1}\|_2^4}{\|(I - \mathbb{P}_V)a_{t+1}\|_2^2} = \|(I - \mathbb{P}_V)a_{t+1}\|_2^2 = d(a_{t+1}, V)^2.$$

We also have

$$\begin{aligned}\|A_{S_t}x^*\|_2^2 = \frac{\|A_{S_t}(I - \mathbb{P}_V)a_{t+1}\|_2^2}{\|(I - \mathbb{P}_V)a_{t+1}\|_2^2} &\le \frac{\|A_{S_t}(I - \mathbb{P}_V)\|_{\mathrm{F}}^2 \|(I - \mathbb{P}_V)a_{t+1}\|_2^2}{\|(I - \mathbb{P}_V)a_{t+1}\|_2^2} \\ &\le \|A_{S_t}(I - \mathbb{P}_V)\|_{\mathrm{F}}^2 = \sum_{i \in S_t} d(a_i, V)^2.\end{aligned}$$

Additionally, when processing the row a_{t+1}, the value of λ used by the algorithm is $\|A_{S_t} - [A_{S_t}]_k\|_{\mathrm{F}}^2/k < \|A_{S_t}(I - \mathbb{P}_V)\|_{\mathrm{F}}^2/k$ since the subspace V has a dimension k. Now, we consider two cases:

- **Case 1:** $\lambda = 0$. In this case, we have $\operatorname{rank}(A_{S_t}) \le k$. There are again two cases. If $a_{t+1} \notin \operatorname{rowspace}(A_{S_t})$, then the algorithm adds $t+1$ to the set S and we are done.
  If $a_{t+1} \in \operatorname{rowspace}(A_{S_t})$, then we can write $a_{t+1} = (A_{S_t})^{\mathrm{T}} z$ for some z. If $A_{S_t}x^* = 0$, then we get $\langle x^*, a_{t+1} \rangle = (x^*)^{\mathrm{T}}(A_{S_t})^{\mathrm{T}} z = \langle z, A_{S_t}x^* \rangle = 0$ which contradicts our assumption that $|\langle a_{t+1}, x^* \rangle|^2 = d(a_{t+1}, V)^2 > \sum_{i \in S_t} d(a_i, V)^2 > 0$. Thus, $A_{S_t}x^* \ne 0$ and therefore

  $$\frac{|\langle a_{t+1}, x^* \rangle|^2}{\|A_{S_t}x^*\|_2^2} \ge \frac{d(a_{t+1}, V)^2}{\sum_{i \in S_t} d(a_i, V)^2} > 1.$$

  Finally, since $a_{t+1} \in \operatorname{rowspace}(A_{S_t})$, we obtain $a_{t+1}^{\mathrm{T}}(A_{S_t}^{\mathrm{T}} A_{S_t})^+ a_{t+1} > 1$ and therefore the algorithm adds $t+1$ to the set S and we are done.
- **Case 2:** $\lambda \ne 0$. In this case, we have $\operatorname{rank}(A_{S_t}) > k$ and therefore $\sum_{i \in S_t} d(a_i, V)^2 > 0$. Now,

  $$\frac{|\langle a_{t+1}, x^* \rangle|^2}{\|A_{S_t}x^*\|_2^2 + \lambda\|x^*\|_2^2} \ge \frac{d(a_{t+1}, V)^2}{\sum_{i \in S_t} d(a_i, V)^2 + \lambda} \ge \frac{\sum_{i \in S_t} d(a_i, V)^2}{\sum_{i \in S_t} d(a_i, V)^2 + \sum_{i \in S_t} d(a_i, V)^2/k} = \frac{1}{1 + 1/k}.$$

  From the above inequality, we obtain $(a_{t+1})^{\mathrm{T}}((A_{S_t})^{\mathrm{T}} A_{S_t} + \lambda I)^+ a_{t+1} > 1/(1 + 1/k)$ and therefore the algorithm adds $t+1$ to the set S and we are done. □

The above lemma now directly implies the following from our earlier discussion:

Lemma 11.3.2. *Let S be the set returned by Algorithm 11.1 after processing the rows $a_1, \ldots, a_n$. For any k-dimensional subspace V,*

$$\max_{i \in S} d(a_i, V) \le \max_{i \in [n]} d(a_i, V) \le \sqrt{|S|} \cdot \max_{i \in S} d(a_i, V).$$

Thus the set S returned by the algorithm is a *strong* coreset with a distortion bounded by $\sqrt{|S|}$.

Hence, if we show that $|S|$ is small, then we obtain the two desired properties of a coreset: (i) the distortion of A_S is small and (ii) the number of rows in A_S is small.

To bound the size of the set S, we use the fact that the online rank-k ridge leverage scores of *all* the rows in the matrix A_S *with respect to* A_S are at least $1/(1+1/k)$. Thus, the number of rows in A_S is at most $1 + 1/k$ times the sum of online rank-k ridge leverage scores of the matrix A_S. We shall now prove a bound on the sum of online rank-k ridge leverage scores of an arbitrary matrix B. The proof of this lemma is similar to that of proof of Lemma 2.11 of [BDM+20]. First, we define a "online rank-k condition number" that we use to bound the sum of online rank-k ridge leverage scores.

Definition 11.3.3 (Online Rank-k Condition Number). Given a matrix B with rows $b_1, \dots, b_n$, let i^* be the largest index i such that $\text{rank}(B_{1:i}) = k$. The online rank-k condition number of B is defined as

$$\kappa \coloneqq \frac{\|B\|_2}{\min_{i \le i^*+1} \sigma_{\min}(B_{1:i})}$$

where $\sigma_{\min}(\cdot)$ denotes the smallest *nonzero* singular value.

Lemma 11.3.4 (Sum of online rank-k ridge leverage scores). *Let $B \in \mathbb{R}^{n \times d}$ be an arbitrary matrix with an online rank-k condition number κ, then*

$$\sum_{i=1}^{n} \tau_i^{\mathrm{OL},k}(B) = O(k \log(k \cdot \kappa)^2).$$

As the proof is largely similar to that of [BDM+20], we defer the proof to the appendix. Applying the above lemma to the matrix A_S, we obtain that $|S| = O(k \cdot \log(k \cdot \kappa(A_S))^2)$. Using the strong coreset property of the matrix A_S, we can show that $\kappa(A_S) \le \sqrt{n} \cdot \kappa(A)$ thereby showing that the coreset has a size at most $|S| = O(k \log(n \cdot \kappa(A))^2)$ and has a distortion at most $O(\sqrt{k} \log(n \cdot \kappa(A)))$. Thus giving the following theorem:

Theorem 11.3.5. *Given rows of any arbitrary $n \times d$ matrix A with an online rank-k condition number κ, Algorithm 11.1 selects a subset S of size $|S| \le O(k(\log n\kappa)^2)$ such that for any k dimensional subspace V, we have*

$$\max_{i \in S} d(a_i, V) \le \max_{i \in [n]} d(a_i, V) \le C\sqrt{k} \cdot \log(n\kappa) \max_{i \in S} d(a_i, V)$$

for a large enough constant C. Additionally, the space requirement of the algorithm is bounded by the amount of space required to store $O(|S|)$ rows of A.

If we assume that all the rows of A lie in a euclidean ball of radius R and that we are given some $\delta < \Delta \coloneqq \min_{k\text{-dim } V} \max_i d(a_i, V)$, then we can obtain bounds on $|S|$ that are independent of n and only depend on the "aspect ratio" R/δ. A similar aspect ratio has been used in an earlier work of Makarychev, Manoj, and Ovsiankin [MMO22]. Let t be a parameter we fix later. We simply feed the vectors $(\delta/t)e_1, \dots, (\delta/t)e_{k+1}$ to Algorithm 11.1 before processing the vectors $a_1, \dots, a_n$. We note that the algorithm is guaranteed to select the vectors $(\delta/t)e_1, \ldots, (\delta/t)e_{k+1}$ since each of these vectors do not lie in the rowspan of the previous vectors. Let S denote the subset of rows of A selected by this algorithm. Using (11.1), we note that for any k-dimensional subspace V,

$$\max_{i \in [n]} d(a_i, V) + \max_{i \in [k+1]} d((\delta/t)e_i, V)$$
$$\leq \sqrt{\sum_{i=1}^{k+1} d((\delta/t)e_i, V)^2 + \sum_{i \in S} d(a_i, V)^2}$$

which implies that

$$\max_{i \in [n]} d(a_i, V) \leq \sqrt{k+1}\frac{\delta}{t} + \sqrt{\sum_{i \in S} d(a_i, V)^2}.$$

We now note that the online rank-k condition number of the coreset computed by the algorithm must be bounded by Rt/δ since the first $k+1$ rows of the coreset are guaranteed to be the points $(\delta/t)e_1, \ldots, (\delta/t)e_{k+1}$. Thus, using Lemma 11.3.4 we obtain $|S| = O(k \log(t|S|R/\delta)^2)$ which implies $|S| \leq O(k \log(kt \cdot R/\delta)^3)$. If we pick $t = 2\sqrt{k+1}$, we obtain the following theorem.

Theorem 11.3.6. *Given that $\delta < \max_{k\text{-dim } V} \max_i d(a_i, V)$ and $\|a_i\|_2 < R$, then we can compute a subset of rows A_S of A such that for any k dimensional subspace V,*

$$\max_i d(a_i, V) \leq C\sqrt{k}(\log kR/\delta)^{3/2} \max_{i \in S} d(a_i, V)$$

and $|S| = O(k \cdot (\log kR/\delta)^3)$. The space requirement of the algorithm is bounded by the amount of space required to store $O(|S|)$ rows of the matrix A.

A coreset S of size $|S|$ and a distortion β can also be used to quickly compute an approximate solution to the ℓ_∞ subspace approximation problem as follows. Let V^* be the optimal solution for the ℓ_∞ subspace approximation problem on A and $\widetilde{V}$ denote the top-k singular subspace of the coreset A_S, which can be computed using the singular value decomposition. Then,

$$\max_i d(a_i, \widetilde{V}) \leq \beta \cdot \max_{i \in S} d(a_i, \widetilde{V}) \leq \beta\sqrt{\sum_{i \in S} d(a_i, \widetilde{V})^2}.$$

Since, $\widetilde{V}$ is the top-k singular subspace of the coreset A_S, we have $\sqrt{\sum_{i \in S} d(a_i, \widetilde{V})^2} \leq \sqrt{\sum_{i \in S} d(a_i, V^*)^2}$ which overall implies

$$\max_i d(a_i, \widetilde{V}) \leq \beta\sqrt{\sum_{i \in S} d(a_i, V^*)^2} \leq \beta\sqrt{|S|} \max_i d(a_i, V^*).$$

Hence, a $\beta\sqrt{|S|}$ approximation to the ℓ_∞ subspace approximation[3] problem can be obtained without using any SDP based algorithms from previous works. We can additionally initialize an alternating minimization algorithm on the coreset for ℓ_∞ subspace approximation using the SVD subspace of the coreset and use convex optimization solvers to further improve the quality of the solution. We do note that there are no known bounds on the solution quality attained by the alternating minimization algorithm.

By a simple (lossy) reduction of outer $(d-k)$ radius estimation problem to computing optimal ℓ_∞ subspace approximation of the matrix $B = A - a_1$ i.e., the matrix obtained by subtracting a_1 from each row of A, we obtain the following theorem using the coreset bounds in Theorem 11.3.6.

Theorem 11.3.7 (Outer $(d-k)$ radius estimation)**.** *Given $0 = a_1 - a_1, \ldots, a_n - a_1$, if a streaming algorithm computes a coreset S with a distortion β, then the outer $(d-k)$ radius of the point set S is an $O(\beta)$ approximation to the outer $(d-k)$ radius of the entire point set.*

Given that the online rank-k condition number of the matrix $A - a_1$ is κ', the outer $(d-k)$ radius of the point set can be approximated up to $\sqrt{k} \cdot \log n\kappa'$ factor by computing the outer $(d-k)$ radius of the coreset points.

Proof. If V is a k-dimensional subspace and c is arbitrary, then the set $V+c$ is defined as a k-dimensional flat. Recall that the outer $d-k$ radius of a point set $\{ a_1, \ldots, a_n \} \subseteq \mathbb{R}^d$ is defined as

$$\min_{k\text{-dim flat } F} \max_i d(a_i, F).$$

Using the fact that flats are translations of k dimensional subspaces, we equivalently have that the outer $d-k$ radius is equal to

$$\min_{k\text{-dim subspace } V} \min_{c \in \mathbb{R}^d} \max_i d(a_i - c, V) = \min_{k\text{-dim subspace } V} \min_c \|(A-c)(I - \mathbb{P}_V)\|_{\infty,2}.$$

Here we abuse the notation and use $A - c$ to denote the matrix with rows given by $a_i - c$ for $i \in [n]$. Now define a matrix $B \doteq A - a_1$ with n rows given by $0 = a_1 - a_1, a_2 - a_1, a_3 - a_2, \ldots, a_n - a_1$. For any k-dimensional subspace V and any $c \in \mathbb{R}^d$, we have

$$\begin{aligned}
\|B(I - \mathbb{P}_V)\|_{\infty,2} &= \|(A - a_1)(I - \mathbb{P}_V)\|_{\infty,2} = \|(A - c + c - a_1)(I - \mathbb{P}_V)\|_{\infty,2} \\
&\le \|(A-c)(I - \mathbb{P}_V)\|_{\infty,2} + \|(I - \mathbb{P}_V)(a_1 - c)\|_2 \\
&\le 2\|(A-c)(I - \mathbb{P}_V)\|_{\infty,2}.
\end{aligned}$$

Hence, $\|B(I - \mathbb{P}_V)\|_{\infty,2} \le 2\min_c \|(A-c)(I - \mathbb{P}_V)\|_{\infty,2}$. We also have $\|B(I - \mathbb{P}_V)\|_{\infty,2} = \|(A - a_1)(I - \mathbb{P}_V)\|_{\infty,2} \ge \min_c \|(A-c)(I - \mathbb{P}_V)\|_{\infty,2}$. Thus, $\min_V \|B(I - \mathbb{P}_V)\|_{\infty,2}$ is a 2-approximation for $\min_{k\text{-dim flat } F} \max_i d(a_i, F)$ and if S is the set of rows selected by Algorithm 11.1 when run on the

[3] In our case, the approximation factor is $O(k(\log n\kappa)^2)$.

rows of the matrix $B = A - a_1$, then

$$\min_V \|B_S(I - \mathbb{P}_V)\|_{\infty,2}$$

is an $O(\sqrt{k}\log(n\kappa'))$ approximation for outer $(d-k)$-radius estimation of the point set $\{ a_1, \dots, a_n \}$ where κ' is the online rank-k condition number of $A - a_1$. $\square$

11.3.3 Fast Implementation of Algorithm 11.1

Note that the set S and hence the value λ are updated only at most $O(k\log(n \cdot \kappa)^2)$ times in the stream. Hence, if we compute the singular value decomposition of A_S each time S is updated, we only spend at most $O(d\,\mathrm{poly}(k, \log n\kappa))$ time in total. Let $U\Sigma V^{\mathrm{T}} = A_S$ be the "thin" singular value decomposition of A_S. Then given any vector a, we can compute $a^{\mathrm{T}}(A_S^{\mathrm{T}}A_S + \lambda I)^{+}a$ as $\|\Sigma^{-1}V^{\mathrm{T}}a\|_2^2 + (1/\lambda)\|(I - VV^{\mathrm{T}})a\|_2^2 = \|Ma\|_2^2$ where M is defined as the matrix obtained by concatenating $\Sigma^{-1}V^{\mathrm{T}}$ and $(1/\sqrt{\lambda})(I - VV^{\mathrm{T}})$.

Now, if $\boldsymbol{G}$ is a Gaussian matrix with $O(\log n)$ rows, we can approximate $\|Ma_i\|_2^2$ with $\|\boldsymbol{G}Ma_i\|_2^2$ up to constant factors for all the *future* rows a_i. Thus, if each time S is updated, we compute the matrix M and sample a Gaussian matrix $\boldsymbol{G}$ and then compute $\boldsymbol{G}M$ which has $O(\log n)$ rows. Then the online rank-k ridge leverage score of any row a_i that appears in the stream can be approximated as $\|(\boldsymbol{G}M)a_i\|_2^2$ in time $O(\mathrm{nnz}(a_i)\log n)$ time since the matrix $\boldsymbol{G}M$ has only $O(\log n)$ rows. Thus, the overall algorithm can be implemented in time $O(\mathrm{nnz}(A)\log n + d \cdot \mathrm{poly}(k, \log n\kappa))$. We implement this algorithm and find that it runs very fast on large datasets.

11.4 Lower Bounds

The algorithm in previous section uses $O(dk(\log n\kappa)^2)$ bits of space to process a stream of n rows in $\mathbb{R}^d$ and outputs a strong coreset with a distortion at most $O(C\sqrt{k}\log n\kappa)$, where κ is the condition number. We show that any algorithm that constructs a strong coreset with distortion $O(\sqrt{k/\log n})$ must use $\Omega(n)$ bits of space. This shows that our algorithm essentially obtains the best possible distortion bounds up to $\mathrm{poly}(\log n\kappa)$ factors. Our argument is similar to that of [WY22]. We state the lower bound in the following theorem.

Theorem 11.4.1. *Given parameters n, d and k with $k = \Omega(\log n)$, any streaming algorithm that computes a strong coreset with distortion at most $O(\sqrt{k/\log n})$ with probability $\geq 9/10$ must use $\Omega(n)$ bits of space.*

Proof. Let n, d and k be arbitrary. Let $a_1, \dots, a_{2n} \in \mathbb{R}^d$ be random vectors sampled as follows: each of the first k entries of each a_i is set to $+1/-1$ with equal probability. The remaining $d - k$ coordinates of each a_i are set to 0.

Note that $\|a_i\|_2^2 = k$ for all i. For arbitrary $i \neq j$, consider $|\langle a_i, a_j \rangle|$. By Hoeffding's inequality, with probability $\ge 1 - \delta$, $|\langle a_i, a_j \rangle| \le O(\sqrt{k \log 1/\delta})$. Setting $\delta = 1/10n^2$ and using a union bound, we obtain that with probability $\ge 9/10$, for all $i \neq j$, $|\langle a_i, a_j \rangle| \le O(\sqrt{k \log n})$. Condition on this event. Let $S \subseteq [2n]$, $|S| = n$ be a uniformly random subset of $[2n]$ of size n.

Consider the stream of vectors $(a_i)_{i \in S}$. Let $\mathscr{C}$ be a randomized algorithm that computes a strong coreset with distortion $\alpha \le O(\sqrt{k/\log n})$ with probability $\ge 9/10$. Let $\mathscr{C}((a_i)_{i\in S})$ be the output of the algorithm $\mathscr{C}$ on the stream $(a_i)_{i\in S}$. Condition on the event that $\mathscr{C}((a_i)_{i\in S})$ is a strong coreset. We now argue that if α is not too large, we can compute the set S from the coreset $\mathscr{C}((a_i)_{i\in S})$.

Given a strong coreset M with distortion α for the stream $(a_i)_{i\in S}$ and a rank-k subspace V, let $M(V)$ be the value computed using the coreset such that

$$M(V) \le \max_{i \in S} d(a_i, V) \le \alpha \cdot M(V).$$

For each $i \in [2n]$, consider the subspace

$$V_i = \operatorname{span}(e_1, \dots, e_k) \cap a_i^{\perp}$$

where $a_i^{\perp}$ denotes the subspace orthogonal to the vector a_i. We now note the following:

- $d(a_i, V_i) = \|a_i\|_2 = \sqrt{k}$
- For all $j \neq i$,
$$d(a_j, V_i) = |\langle a_j, a_i \rangle| / \|a_i\|_2 \le O(\sqrt{\log n}).$$

Therefore, if $i \in S$, then $\mathscr{C}((a_j)_{j\in S})(V_i) \ge \sqrt{k}/\alpha$ and if $i \notin S$, then $\mathscr{C}((a_j)_{j\in S})(V_i) \le O(\sqrt{\log n})$. If the distortion $\alpha \le \sqrt{k/\log n}$, then enumerating over all V_i for $i \in [2n]$ and computing $\mathscr{C}((a_j)_{j\in S})(V_i)$, we can determine the set S.

Let S' be the set computed by the enumeration algorithm. If $|S'| \neq n$, set S' to $\{1, 2, \dots, n\}$. By the above discussion, we have $\mathbf{Pr}[S' = S] \ge 9/10$. Note that the entropy of the set S is $t = \Omega(n)$ where $2^t = \binom{2n}{n}$ is the number of subsets of $[2n]$ of size $[n]$.

We now upper bound the conditional entropy $H(S' \mid S)$. Let I denote the indicator random variable denoting if the coreset construction algorithm succeeds. Note that given $I = 1$, we have $S = S'$. We have

$$\begin{aligned} H((S, S')) &= H(S) + I(S\,;\,S') \\ \text{and} \quad H((S, S')) &\le H((S, S', I)) = H(S) + H(I \mid S) + H(S' \mid I, S) \end{aligned}$$

and therefore, $I(S\,;\,S') \le H(I \mid S) + H(S' \mid I, S)$. Since we assumed that the coreset construction algorithm succeeds with probability $\ge 9/10$ given any instance, we have $H(I \mid S) \le (9/10)\log_2(10/9)+$

$(1/10)\log_2(10) \le 1/2$. Now,

$$\begin{aligned}
&H(\boldsymbol{S}' \mid \boldsymbol{I}, \boldsymbol{S})\\
&= \sum_S \mathbf{Pr}[\boldsymbol{S} = S] \cdot [H(\boldsymbol{S}' \mid \boldsymbol{S} = S, \boldsymbol{I} = 0) \cdot \mathbf{Pr}[\boldsymbol{I} = 0 \mid \boldsymbol{S} = S] + H(\boldsymbol{S}' \mid \boldsymbol{S} = S, \boldsymbol{I} = 1) \cdot \mathbf{Pr}[\boldsymbol{I} = 1 \mid \boldsymbol{S} = S]]\\
&\le \sum_S \mathbf{Pr}[\boldsymbol{S} = S] \cdot H(\boldsymbol{S}' \mid \boldsymbol{S} = S, \boldsymbol{I} = 0) \cdot (1/10)
\end{aligned}$$

where we used the fact that if $\boldsymbol{I} = 1$, then $\boldsymbol{S}' = \boldsymbol{S}$ and therefore $H(\boldsymbol{S}' \mid \boldsymbol{S} = S, \boldsymbol{I} = 1) = 0$. Since the output $\boldsymbol{S}'$ is always a subset of $[2n]$ of size n, we have $H(\boldsymbol{S}' \mid \boldsymbol{S} = S, \boldsymbol{I} = 0) \le \log_2 \binom{2n}{n} = t$ which then implies

$$H(\boldsymbol{S}' \mid I, \boldsymbol{S}) \le t/10.$$

Hence, the mutual information $I(\boldsymbol{S}\,;\,\boldsymbol{S}') \ge 9t/10 - 1/2$ and by the data processing inequality, we have

$$I(\mathscr{C}((a_i)_{i\in\boldsymbol{S}})\,;\,\boldsymbol{S}) \ge 9t/10 - 1/2 \tag{11.2}$$

which implies that the space necessary to store the coreset is $\Omega(n)$ bits since $t = \log_2 \binom{2n}{n} = \Omega(n)$. □

11.5 ℓ_p Subspace Approximation

We now show that our coreset construction algorithm for the ℓ_∞ subspace approximation problem, extends to the ℓ_p subspace approximation problem. Fix a matrix A. For any k-dimensional subspace V, let d_V denote the nonnegative vector satisfying $(d_V)_i = \text{dist}(a_i, V) = \|a_i^{\mathrm{T}}(I - \mathbb{P}_V)\|_2$. Hence, the ℓ_p subspace approximation problem is to find the rank-k subspace V that minimizes $\|d_V\|_p$. We use exponential random variables to embed ℓ_p low rank approximation problem into an ℓ_∞ low rank approximation problem. We then use the coreset construction algorithm for ℓ_∞ LRA to obtain a coreset for the ℓ_p LRA. First, we have the following lemma about exponential random variables that has been used in various previous works to embed ℓ_p problems into an ℓ_∞ problem.

Lemma 11.5.1. *Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ be independent exponential random variables. Then with probability $\ge 1-\delta$, $\max_i \boldsymbol{e}_i^{-1/p}|x_i| \ge \|x\|_p/(\log 1/\delta)^{1/p}$. We also have that with probability $\ge 1 - \delta$, $\max_i \boldsymbol{e}_i^{-1/p}|x_i| \le \delta^{-1/p} \cdot \|x\|_p$.*

Proof. By min-stability of exponential random variables, we have that the distribution of $\max_i \boldsymbol{e}^{-1}|x_i|^p$ is the same as the distribution of $\boldsymbol{e}^{-1}\|x\|_p^p$ where $\boldsymbol{e}$ is also a standard exponential random variable.

With probability $\geq 1-\delta$, we have $\boldsymbol{e} \leq \log 1/\delta$. And hence we have that with probability $\geq 1-\delta$,

$$\max_i \boldsymbol{e}_i^{-1/p}|x_i| = (\max_i \boldsymbol{e}_i^{-1}|x_i|^p)^{1/p} \geq \frac{\|x\|_p}{(\log 1/\delta)^{1/p}}.$$

With probability $\geq 1-\delta$, we also have that $\boldsymbol{e} \geq \delta$ which implies that with probability $\geq 1-\delta$, $\max_i \boldsymbol{e}_i^{-1/p}|x_i| = (\max_i \boldsymbol{e}_i^{-1}|x_i|^p)^{1/p} \leq \|x\|_p \delta^{-1/p}$. □

Given n, define $\boldsymbol{D}$ to be a random matrix with diagonal entries given by independent copies of the random variable $\boldsymbol{e}^{-1/p}$. For any fixed rank k projection matrix P, the above lemma implies that $\|\boldsymbol{D}A(I-P)\|_{\infty,2} \geq \|A(I-P)\|_{p,2}/(\log 1/\delta)^{1/p}$. But we can not union bound over the net of all k dimensional subspace of $\mathbb{R}^d$ since the net can have as many as $\exp(dk)$ subspaces which leads to a distortion of $d^{1/p}$ which is prohibitive. Here we crucially use the fact that Algorithm 11.1 only selects a coreset with $m = O(k \cdot (\log n\kappa)^2)$ rows. So only those k dimensional subspaces spanned by at most m rows of A are of interest to us. Now, we can union bound over a net of $\exp(\text{poly}(k, \log n\kappa))$ subspaces and show the following lemma:

Lemma 11.5.2. *Let $\boldsymbol{D}$ be an $n \times n$ diagonal matrix with each diagonal entry being an independent copy of the random variable $\lceil \boldsymbol{e}^{-1/p} \rceil$. Fix an $n \times d$ matrix A. With probability $\geq 98/100$, for all k dimensional subspaces that are in the span of at most $m = O(k \log^2 n\kappa)$ rows of A, we have,*

$$\|\boldsymbol{D} \cdot d_V\|_\infty \geq \|d_V\|_p / 2(\log 100 + m \log n + km \log n\kappa)^{1/p}.$$

Proof. Let S be an arbitrary set of $m \leq K$ rows of A and let $V_S \coloneqq \text{rowspace}(A_S)$. Let N_S be a γ net for the set $V_S \cap \mathbb{S}^{d-1}$ i.e., the set of vectors in the subspace V_S with euclidean norm 1. As the subspace V_S has dimension at most m, we have that there is a set N_S with size at most $\exp(O(m \log 1/\gamma))$. Let V be an arbitrary k dimensional subspace of V_S and let $\{v_1, \ldots, v_k\}$ be an orthonormal basis for V.

Let $\widetilde{V}$ be the subspace spanned by $\{\widetilde{v}_1, \ldots, \widetilde{v}_k\}$, where $\widetilde{v}_i \in N_S$ and $\|v_i - \widetilde{v}_i\|_2 < \gamma$ for all $i \in [n]$. Let a be an arbitrary vector. By abusing the notation let V (resp. $\widetilde{V}$) also denote the matrix with $v_1, \ldots, v_k$ (resp. $\widetilde{v}_1, \ldots, \widetilde{v}_k$) as columns. We have

$$d(a, V) = \|a - VV^{\mathrm{T}}a\|_2 \quad \text{and} \quad d(a, \widetilde{V}) = \|a - \widetilde{V}\widetilde{V}^{+}a\|_2$$

and therefore $|d(a,V) - d(a,\widetilde{V})| \leq \|\widetilde{V}\widetilde{V}^{+} - VV^{\mathrm{T}}\|_2\|a\|_2$. If $\gamma \leq 1/4\sqrt{k}$, we can show that $\|VV^{\mathrm{T}} - \widetilde{V}\widetilde{V}^{+}\|_2 \leq 4\sqrt{k}\gamma$ and therefore have that for any a, $|d(a,V) - d(a,\widetilde{V})| \leq \sqrt{k}\gamma\|a\|_2$. Hence,

$$\|d_V - d_{\widetilde{V}}\|_\infty \leq \max_i |d(a_i, V) - d(a_i, \widetilde{V})| \leq 4\sqrt{k}\gamma \max_i \|a_i\|_2 = 4\sqrt{k}\gamma\|A\|_{\infty,2}.$$

Overall, this implies that for any arbitrary k dimensional subspace V in the span of rows of A_S, there is a k dimensional subspace $\widetilde{V}$ spanned by some k vectors in the net N_S satisfying

$$\|d_V - d_{\widetilde{V}}\|_\infty \le 4\sqrt{k}\gamma\|A\|_{\infty,2}.$$

As $d_V \in \mathbb{R}^n$, we have $\|d_V - d_{\widetilde{V}}\|_p \le n^{1/p}\|d_V - d_{\widetilde{V}}\|_\infty \le 4\sqrt{k}\gamma n^{1/p}\|A\|_{\infty,2}$. Now, let

$$\mathscr{V}_S \coloneqq \{\widetilde{V} = \operatorname{span}(\widetilde{v}_1, \ldots, \widetilde{v}_k) \mid \widetilde{v}_i \in N_S\}.$$

We have $|\mathscr{V}_S| \le |N_S|^k \le \exp(O(km\log 1/\gamma))$ since $|N_s| \le \exp(O(m\log 1/\gamma))$. As there are $\binom{n}{m}$ choices for S, the total number of subspaces in the set $\cup_{S \in \binom{[n]}{m}} \mathscr{V}_S$ is upper bounded by $\exp(m\log n + km\log 1/\gamma)$. Using Lemma 11.5.1, using a union bound over all $\exp(m\log n + km\log 1/\gamma)$ choices of $\widetilde{V}$, we have that with probability $\ge 99/100$, for all $\widetilde{V} \in \cup_{\binom{[n]}{m}} V_S$,

$$\|\boldsymbol{D} \cdot d_{\widetilde{V}}\|_\infty \ge \frac{\|d_{\widetilde{V}}\|_p}{(\log 100 + m\log n + km\log 1/\gamma)^{1/p}}.$$

Using Lemma 11.5.1 again, we also have that $\max_i |\boldsymbol{D}_i| \le C_3 n^{1/p}$ for a large enough constant C_3 with probability $\ge 99/100$. Condition on both these events. We have that for any k dimensional subspace V in the span of any set of m rows of A,

$$\begin{aligned}\|\boldsymbol{D} \cdot d_V\|_\infty &\ge \|\boldsymbol{D} \cdot d_{\widetilde{V}}\|_\infty - \|\boldsymbol{D} \cdot (d_V - d_{\widetilde{V}})\|_\infty \\ &\ge \frac{\|d_{\widetilde{V}}\|_p}{(\log 100 + m\log n + km\log 1/\gamma)^{1/p}} - C_1 n^{1/p}\|d_V - d_{\widetilde{V}}\|_\infty \\ &\ge \frac{\|d_V\|_p}{(\log 100 + m\log n + km\log 1/\gamma)^{1/p}} - \frac{4\sqrt{k}n^{1/p}\gamma\|A\|_{\infty,2}}{(\log 100 + m\log n + km\log 1/\gamma)^{1/p}} \\ &\quad - 4C_1 n^{1/p}\sqrt{k}\gamma\|A\|_{\infty,2}.\end{aligned}$$

For any V, we have that $\|d_V\|_p \ge \|d_V\|_2/\sqrt{n} \ge \|A - [A]_k\|_{\mathrm{F}}/\sqrt{n}$ using the fact that V is a k dimensional subspace. Hence, if $\gamma \le \operatorname{poly}(\|A - [A]_k\|_{\mathrm{F}}/\|A\|_{\infty,2}, 1/n)$, then

$$\|\boldsymbol{D} \cdot d_V\|_\infty \ge \frac{\|d_V\|_p}{2(\log 100 + m\log n + km\log 1/\gamma)^{1/p}}.$$

Now, γ can be taken as $\operatorname{poly}(1/(n\kappa))$ so that

$$\|\boldsymbol{D} \cdot d_V\|_\infty \ge \frac{\|d_V\|_p}{C(\log 100 + m\log n + km\log(n\kappa))^{1/p}}$$

for all subspaces V that are in the span of any subset of m rows of A. □

If V^* is the optimal solution for the ℓ_p subspace approximation problem, we can also condition on the event that $\|\boldsymbol{D} \cdot d_{V^*}\|_\infty \le C\|d_{V^*}\|_p$ for a large enough constant C. We can now argue that if S is the subset of rows selected by Algorithm 11.1 when run on the matrix $\boldsymbol{D}A$, if $\hat{V}$ is an approximate solution for the ℓ_∞ subspace approximation problem on the points $(\boldsymbol{D}A)_S$, then $\hat{V}$ is also a good solution for the ℓ_p subspace approximation problem of A.

Theorem 11.5.3. *Let $\boldsymbol{D}$ be an $n \times n$ random matrix with each diagonal entry being an independent copy of $\lceil \boldsymbol{e}^{-1/p} \rceil$ where $\boldsymbol{e}$ is a standard exponential random variable. If S is the subset selected by Algorithm 11.1 when run on the rows of the matrix $\boldsymbol{D} \cdot A$ and if $\hat{V}$ is a β approximate solution to the problem $\min_{k\text{-dim } V} \|(\boldsymbol{D}A)_S(I - \mathbb{P}_V)\|_{\infty,2}$, then with probability $\ge 9/10$,*

$$\frac{\|A(I - \mathbb{P}_{\hat{V}})\|_{p,2}}{\min_{k\text{-dim } V} \|A(I - \mathbb{P}_V)\|_{p,2}} \le \beta \cdot O(k^{1/2+2/p} \log^{1+3/p} n\kappa).$$

Proof. Let

$$V^* = \operatorname*{arg\,min}_{k\text{-dim subspaces } V} \|d_V\|_p.$$

Condition on the event that $\|\boldsymbol{D}^{1/p} \cdot d_{V^*}\|_\infty \le C_1\|d_{V^*}\|_p$ for a large enough constant C_1. The event holds with probability $\ge 99/100$ by Lemma 11.5.1. Finally, by a union bound, we have all the following events hold simultaneously with probability $\ge 9/10$:

1. Algorithm 11.1, when run on the rows of the matrix $\boldsymbol{D} \cdot A$, selects at most $m = O(k \cdot (\log n\kappa)^2)$ rows.
2. For any k dimensional subspace V contained in the span of any at most m rows of A,
$$\|\boldsymbol{D} \cdot d_V\|_\infty \ge \frac{\|d_V\|_p}{C_2 k^{2/p} \log^{3/p} n\kappa}.$$
3. If V^* is the optimal subspace that minimizes the ℓ_p norm of the distance vector to a k dimensional subspace, then
$$\|\boldsymbol{D} \cdot d_{V^*}\|_\infty \le C_1\|d_{V^*}\|_p.$$

Conditioned on the above events, let $S \subseteq [n]$ be the coreset computed for the matrix $\boldsymbol{D} \cdot A$ by Algorithm 11.1. From Theorem 11.3.5, we have that for any rank k projection matrix P,

$$\|(\boldsymbol{D}A)_S(I - P)\|_{\infty,2} \le \|(\boldsymbol{D}A)(I - P)\|_{\infty,2} \le C\sqrt{k}(\log n\kappa)\|(\boldsymbol{D}A)_S(I - P)\|_{\infty,2}.$$

Let $\hat{V}$ be a k dimensional subspace such that

$$\min_{k\text{-dim } V} \|(\boldsymbol{D}A)_S(I - \mathbb{P}_{\hat{V}})\|_{\infty,2}\beta \cdot \min_{k\text{-dim } V} \|(\boldsymbol{D}A)_S(I - \mathbb{P}_V)\|_{\infty,2}$$

Without loss of generality, we can assume that $\hat{V}$ is contained in the rowspace of $(\boldsymbol{D} \cdot A)_S$ and hence the row space of A_S. Therefore,

$$\begin{aligned}
\|A(I - \mathbb{P}_{\hat{V}})\|_{p,2} = \|d_{\hat{V}}\|_p &\\
&\le C_2 k^{2/p} \log^{3/p}(n\kappa) \|\boldsymbol{D} \cdot d_{\hat{V}}\|_\infty \\
&= C_2 k^{2/p} \log^{3/p}(n\kappa) \|(\boldsymbol{D} \cdot A)(I - \mathbb{P}_{\hat{V}})\|_{\infty,2} \\
&\le C_2 \cdot C \cdot k^{2/p+1/2} \log^{1+3/p}(n\kappa) \|(\boldsymbol{D}A)_S(I - \mathbb{P}_{\hat{V}})\|_{\infty,2} \\
&\le \beta \cdot C_2 \cdot C \cdot k^{2/p+1/2} \log^{1+3/p}(n\kappa) \|(\boldsymbol{D}A)_S(I - \mathbb{P}_{V^*})\|_{\infty,2} \\
&= \beta \cdot C_2 \cdot C \cdot k^{2/p+1/2} \log^{1+3/p}(n\kappa) \|\boldsymbol{D} \cdot d_{V^*}\|_{\infty,2} \\
&\le \beta \cdot C_1 \cdot C_2 \cdot C \cdot k^{2/p+1/2} \log^{1+3/p}(n\kappa) \|d_{V^*}\|_p.
\end{aligned}$$

Thus, $\hat{V}$ is an $O(\beta \cdot k^{2/p+1/2} \log^{1+3/p}(n\kappa))$ approximate solution for the ℓ_p low rank approximation problem over the matrix A. □

11.6 Applications to Other Geometric Streaming Problems

Given a matrix A, suppose that the rows of A are close to a k-dimensional subspace in the following sense: $\Delta \coloneqq \min_{k\text{-dim } V} \max_i d(a_i, V)$ is small. We now show that if S is the subset of rows selected by Algorithm 11.1, then for any vector x, $\|Ax\|_\infty$ can be approximated using $\|A_S x\|_\infty$. Fix any unit vector x. Let i be the index such that $\|Ax\|_\infty = |\langle a_i, x\rangle|$. If $i \in S$, we clearly have $\|Ax\|_\infty = \|A_S x\|_\infty$ and we are done. If $i \notin S$, we obtain that

$$\max_x \frac{|\langle a_i, x\rangle|^2}{\|A_{S<i}x\|_2^2 + \|A_{S<i} - [A_{S<i}]_k\|_{\mathrm{F}}^2/k} \le \frac{1}{1 + 1/k}$$

which implies

$$\begin{aligned}
\|Ax\|_\infty^2 = |\langle a_i, x\rangle|^2 &\le \|A_{S<i}x\|_2^2 + \|A_{S<i} - [A_{S<i}]_k\|_{\mathrm{F}}^2/k \\
&\le \|A_S x\|_2^2 + \|A_S - [A_S]_k\|_{\mathrm{F}}^2/k.
\end{aligned}$$

Let V^* be the optimal solution for rank-k ℓ_∞ subspace approximation of A. We then have, $\|Ax\|_\infty^2 \le \|A_S x\|_2^2 + \|A_S(I - \mathbb{P}_{V^*})\|_{\mathrm{F}}^2/k \le \|A_S x\|_2^2 + |S|\Delta^2/k$. Using $|S| = O(k \log^2 n\kappa)$, we get the following lemma.

Lemma 11.6.1. *If S is the subset of rows selected by Algorithm 11.1, for any k-dimensional subspace U and any unit vector x,*

$$\frac{\|A_S x\|_2}{C\sqrt{k}\log n\kappa} \le \|A_S x\|_\infty \le \|Ax\|_\infty \le \|A_S x\|_2 + C\Delta \log n\kappa.$$

Additionally, as $\|A_S x\|_2 \le \sqrt{|S|}\|A_S x\|_\infty$, we also have

$$\|A_S x\|_\infty \le \|Ax\|_\infty \le (C\sqrt{k}\log n\kappa)\|A_S x\|_\infty + C\Delta \log n\kappa.$$

Width Estimation. Given a point set $a_1, \ldots, a_n \in \mathbb{R}^d$, then the width of the point set in the direction $x \in \mathbb{R}^d$, for a unit vector x is defined as $w(x) \coloneqq \max_i \langle a_i, x\rangle - \min_i \langle a_i, x\rangle$. Using a coreset for estimating $\|Ax\|_\infty$, [WY22] give an $O(\sqrt{d \log n})$ approximation to the width estimation problem. Using Lemma 11.6.1, we show that we get better approximations when Δ is small.

Note that $w(x) = \max_i \langle a_i - a_1, x\rangle - \min_i \langle a_i - a_1, x\rangle$. Now, $\max_i \langle a_i - a_1, x\rangle \ge \langle 0, x\rangle = 0$ and $\min_i \langle a_i - a_1, x\rangle \le \langle 0, x\rangle \le 0$ which implies that $\|(A - a_1)x\|_\infty \le w(x) \le 2\|(A - a_1)x\|_\infty$.

Let κ' be the online rank-k condition number of $A - a_1$. If S is the subset selected by the algorithm when run on the rows $0 = a_1 - a_1, a_2 - a_1, \ldots, a_n - a_1$, then from Lemma 11.6.1, we have $\|(A - a_1)_S x\|_\infty \le \|(A - a_1)x\|_\infty \le w(x)$ and also that $w(x) \le 2\|(A - a_1)x\|_\infty \le 2C\sqrt{k}\log(n\kappa')\|(A - a_1)_S x\|_\infty + 2C\Delta\log(n\kappa')$. Thus, $w'(x) \coloneqq \|(A - a_1)_S x\|_\infty$ satisfies

$$w(x)/2C\sqrt{k}\log(n\kappa') - \Delta/\sqrt{k} \le w'(x) \le w(x)$$

for a large enough constant C. When Δ is very small, for the interesting directions where width is large enough, we obtain a better multiplicative error of $O(\sqrt{k}\log n\kappa')$ as compared to $O(\sqrt{d \log n})$ achieved by the algorithm of [WY22]. Notice that we do not contradict the lower bounds of [AS15] for width estimation because of the additive error that we allow.

Löwner-John Ellipsoid. Given a symmetric convex body, the Löwner-John ellipsoid is defined to be the ellipsoid of minimum volume that encloses the convex body. We consider the case when the convex body is defined as $K = \{x \mid \|Ax\|_\infty \le 1\}$ where the streaming algorithm sees the rows of matrix A one after the other. Woodruff and Yasuda [WY22] show that their coreset can be used to compute an ellipsoid E' such that $E' \subseteq K \subseteq O(\sqrt{d \log n})E'$.

When $k \ll d$, Algorithm 11.1 selects $\ll d$ number of rows and does not have the full d-dimensional *view* of the point set and hence can not compute an ellipsoid that satisfies the above multiplicative definition if the points spans $\mathbb{R}^d$. So we consider the set $K \cap B(0,1)$ and give an algorithm that computes an unbounded ellipsoid E' such that $E' \cap B(0,1) \subseteq K \cap B(0,1) \subseteq (\alpha E') \cap B(0,1)$.

By Lemma 11.6.1, we have that if $\|Ax\|_\infty \le 1$ and $\|x\|_2 = 1$, then $\|A_S x\|_2 \le C\sqrt{k}\log n\kappa$ and if $\|A_S x\|_2 \le 1 - C\Delta\log n\kappa$ and $\|x\|_2 \le 1$, then $\|Ax\|_\infty \le 1$. Now assuming $\Delta < 1/(C\log n\kappa)$, define $E' = \{x \mid \|A_S x\|_2 \le 1 - (C\log n\kappa)\Delta\}$.

From the above, we have that if $x \in E' \cap B(0,1)$, then $x \in K \cap B(0,1)$. Additionally, if $x \in K \cap B(0,1)$, then $\|A_S x\|_2 \le C\sqrt{k}\log n\kappa$ and therefore $x \in \frac{C\sqrt{k}\log n\kappa}{1-(C\Delta\log n\kappa)}E' \cap B(0,1)$. Hence,

$$E' \cap B(0,1) \subseteq K \cap B(0,1) \subseteq \frac{C\sqrt{k}\log n\kappa}{1 - (C\Delta\log n\kappa)}E' \cap B(0,1).$$

11.7 Conclusions and Open Questions

In this chapter, we obtain a deterministic single-pass streaming algorithm, which is very fast in practice, for constructing strong coresets for approximating the maximum distance to any k-dimensional subspace. Using the strong coreset for ℓ_∞ subspace approximation, we obtain a weak coreset for the ℓ_p subspace approximation problem and for other geometric problems such as width estimation.

An interesting open question is if an $O(\sqrt{k}\log n)$ approximation factor can be obtained, removing the dependence on the condition number, when all the coordinates are integers with absolute values bounded by $\text{poly}(n)$. Woodruff and Yasuda [WY22] obtain an $O(\sqrt{d}\cdot\log n)$ approximation for approximating $\|Ax\|_\infty$ for all vectors x using pseudo-determinants. Finding an analogue of pseudo-determinants for rank-k ridge leverage scores may be a path to obtaining $O(\sqrt{k}\log n)$ approximation factor for the ℓ_∞ subspace approximation problem.

Chapter 12

Approximating the Top Eigenvector in Random Order Streams

12.1 Introduction

In this chapter, we consider the problem of approximating the top eigenvector streams. In this problem, we are given vectors $a_1, \ldots, a_n \in \mathbb{R}^d$ one at a time in a stream. Let A be an $n \times d$ matrix with rows $a_1, \ldots, a_n$. The task is to approximate the top eigenvector of the matrix $A^{\mathrm{T}}A$. We use $v_1 \in \mathbb{R}^d$ to denote the top eigenvector of $A^{\mathrm{T}}A$. We focus on obtaining streaming algorithms that use a small amount of space and can output a unit vector $\hat{v}$ such that $\langle \hat{v}, v_1 \rangle^2 \geq 1 - f(R)$, where $f(R)$ is a decreasing function in the gap $R = \lambda_1(A^{\mathrm{T}}A)/\lambda_2(A^{\mathrm{T}}A)$. Here $\lambda_1(\cdot), \lambda_2(\cdot)$ denote the two largest eigenvalues. As the gap R becomes larger, the eigenvector approximation problem becomes easier, and we want more accurate approximations to the eigenvector v_1.

If one is allowed to use $\widetilde{O}(d^2)$[1] bits of space, we can maintain the matrix $A^{\mathrm{T}}A = \sum_i a_i a_i^{\mathrm{T}}$ as we see the rows a_i in the stream, and at the end of processing the stream, we can compute the exact top eigenvector v_1. When the dimension d is large, the requirement of $\Omega(d^2)$ bits of memory can be impractical. Hence, an interesting question is to study non-trivial streaming algorithms that use less memory. In this work, we focus on obtaining algorithms that use $\widetilde{O}(d)$ bits of space.

In the offline setting (where the entire matrix A is available to us), fast iterative algorithms such as [Gu15, MM15, MMS18] can be used to quickly obtain accurate approximations to the top eigenvector when the gap $R = \Omega(1)$. In a single pass streaming setting, we cannot run these algorithms as these iterative algorithms need to *see* the entire matrix multiple times.

There have been two major lines of work studying the problem of eigenvector approximation and the related Principal Component Analysis (PCA) problem in the streaming setting with near-linear in d memory. In the first line of work, each row encountered in the stream is sampled independently from an unknown distribution with mean 0 and covariance Σ and the task is to approximate the top

[1]The notation $\widetilde{O}(f(n))$ is used to denote the set of functions in $O(f(n) \cdot \mathrm{polylog}(n))$.

eigenvector of Σ using the samples. In this line of work, the sample complexity required for algorithms using $O(d \cdot \text{polylog}(d))$ bits of space to output an approximation to v_1, is the main question. The algorithms are usually a variant of Oja's algorithm [Oja82, JJK+16, AZL17, HNWW21, KS24] or the block power method [HP14, BDWY16]. We note that Kumar and Sarkar [KS24] relax the i.i.d. assumption and analyze the sample complexity of Oja's algorithm for estimating the top eigenvector in the Markovian data setting.

The other line of work studies algorithms for arbitrary streams appearing in an arbitrary order. In this setting, we want algorithms to work for *any* input stream given in *any* order. A problem closely related to the eigenvector estimation problem is the Frobenius-norm Low Rank Approximation [CW17, BWZ16, Upa16, GLPW16]. The deterministic Frequent Directions sketch [GLPW16] can, using $\widetilde{O}(d/\varepsilon)$ bits of space, output a unit vector u such that

$$\|A(I - uu^{\mathrm{T}})\|_{\mathrm{F}}^2 \le (1+\varepsilon)\|A(I - v_1 v_1^{\mathrm{T}})\|_{\mathrm{F}}^2.$$

Although the vector u is a $1+\varepsilon$ approximate solution to the Frobenius norm Low Rank Approximation problem, it is possible that the vector u may be (nearly) orthogonal to the top eigenvector v_1. Hence, the Frequent Directions sketch does not guarantee top eigenvector approximation. Recently, Price [Pri23] studies the eigenvector approximation problem in arbitrary streams and obtains results in terms of the gap R of the instance. Price proved that when $R = \Omega(\log n \cdot \log d)$, a variant of Oja's algorithm outputs a unit vector $\hat{v}$ such that

$$\langle \hat{v}, v_1 \rangle^2 \ge 1 - \frac{C \log d}{R} - \frac{1}{\text{poly}(d)}$$

where C is a large enough universal constant. On the lower bound side, Price showed that any algorithm that outputs a vector $\hat{v}$ satisfying

$$\langle \hat{v}, v_1 \rangle^2 \ge 1 - \frac{1}{CR^2},$$

must use $\Omega(d^2/R^3)$ bits of space while processing the stream. This lower bound shows that in the important case of $R = O(1)$, the *correlation*[2] that can be obtained by an algorithm using $\widetilde{O}(d)$ bits of space is at most a constant less than 1. Thus, the current best algorithms for arbitrary streams work only when $R = \Omega(\log n \cdot \log d)$. Thus, for the important case of $R = O(1)$, there are no existing algorithms requiring significantly less than d^2 bits of memory.

We identify an instance with $R = \Theta(\log d/\log\log d)$ where Price's algorithm fails to produce a vector with even a constant correlation with the vector v_1. This shows that Price's algorithm or other variants of Oja's algorithm may fail to extend to the case when $R = O(1)$. We further show that Price's algorithm fails to produce such a vector even when the rows in our hard instance are

[2]We say that the value $\langle u, v \rangle^2$ denotes the correlation between unit vectors u and v.

ordered uniformly at random, showing that even randomly ordered streams can be hard to solve for variants of Oja's algorithm.

In this work, we focus on algorithms that work on worst case inputs A while assuming that the rows of A are *uniformly randomly ordered*. This model is midway between the i.i.d. setting and the arbitrary order stream setting in terms of the generality of streams that can be modeled. We note that a number of works [MP80, GMV05, CCM08, GM09, AS23b] have previously considered streaming algorithms and lower bounds for worst case inputs with random order streams, as it is a natural model often arising in practical settings. Our algorithms are parameterized in terms of the number of **heavy** rows in the stream. We define a row a_i to be *heavy* if $\|a_i\|_2 \geq \|A\|_{\mathrm{F}}/\sqrt{d \cdot \mathrm{polylog}(d)}$. Note that in any stream of rows, by definition, there are at most $O(d \cdot \mathrm{polylog}(d))$ heavy rows. We state our theorem informally below:

Theorem 12.1.1. *Let $a_1, \ldots, a_n \in \mathbb{R}^d$ be a randomly ordered stream and let A denote the $n \times d$ matrix with rows given by $a_1, \ldots, a_n$. If $R = \lambda_1(A^{\mathrm{T}}A)/\lambda_2(A^{\mathrm{T}}A) > C$ for a large enough constant C and the number of heavy rows in the stream is at most h, then there is a streaming algorithm using $O(h \cdot d \cdot \mathrm{polylog}(d))$ bits of space and outputting a unit vector $\hat{v}$ satisfying*

$$\langle \hat{v}, v_1 \rangle^2 \geq 1 - O(1/\sqrt{R})$$

with a probability $\geq 4/5$.

Our algorithm is a variant of the block power method. Along the way, we also improve the gap requirements in the results of Price [Pri23]. We show that by subsampling a stream of rows, Price's algorithm can be made to work even when the gap R is $\Omega(\log^2 d)$ in arbitrary order streams, improving on the $\Omega(\log n \cdot \log d)$ requirement in Price's analysis. We also show that in random order streams, a gap of $\Omega(\log d)$ is sufficient for Price's algorithm, though our algorithm improves on this and works for even a constant gap.

Similar to the lower bound of Price, we show that any algorithm for random order streams must use $\Omega(h \cdot d/R)$ bits of space to output a vector $\hat{v}$ satisfying $\langle \hat{v}, v_1 \rangle^2 \geq 1 - 1/CR^2$ where C is a constant. We summarize the theorem below.

Theorem 12.1.2. *Consider an arbitrary random order stream $a_1, \ldots, a_n$ with the gap parameter $\frac{\sigma_1(A)^2}{\sigma_2(A)^2} = R$. Let h be the number of* heavy *rows in the stream. Any streaming algorithm that outputs a unit vector $\hat{v}$ such that*

$$\langle \hat{v}, v_1 \rangle^2 \geq 1 - 1/CR^2$$

for a large enough constant C, with a probability $\geq 1 - (1/2)^{R+1}$ over the ordering of the stream and its internal randomness, must use $\Omega(h \cdot d/R)$ bits of space.

Techniques. The randomized power method [Gu15] algorithm to approximate the top eigenvector samples a random Gaussian vector $\boldsymbol{g}$ and iteratively computes the vector $v = (A^{\mathrm{T}}A)^t \boldsymbol{g}$[3] for $t = \Theta(\log d)$ iterations and shows that when the gap R is large, $v/\|v\|_2$ is a good approximation for v_1. Thus, the algorithm needs to *see* the quadratic form $A^{\mathrm{T}}A$ multiple times and hence, it cannot be implemented in the single-pass streaming setting considered here.

Assume that the stream is randomly ordered and that there are no heavy rows. Our key observation is that if the stream is long enough, then we can see t approximations $\boldsymbol{B}_j^{\mathrm{T}}\boldsymbol{B}_j$[4] of the quadratic form $A^{\mathrm{T}}A$. Here the matrices $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_t$ are formed by sampling and rescaling the rows of the matrix A and importantly, the rows of $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_t$ do not overlap in the stream, that is, they appear one after the other. Thus, we can compute $v' = (\boldsymbol{B}_t^{\mathrm{T}}\boldsymbol{B}_t) \cdots (\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1) \cdot \boldsymbol{g}$ for the starting vector $\boldsymbol{g}$ in a single pass over the stream. We prove that such matrices $\boldsymbol{B}_j$ exist using the row norm sampling result of [MI10]. Now, the main issue is to show that $v'/\|v'\|_2$ is a good approximation to the top eigenvector v_1. We crucially use a singular value inequality of [WX97] to prove that $\|\boldsymbol{B}_j^{\mathrm{T}}\boldsymbol{B}_j - A^{\mathrm{T}}A\|_2 \le \varepsilon\|A\|_2^2$ for all j suffices for $v'/\|v'\|_2$ to be a good approximation to v_1.

The above analysis assumes that there are no heavy rows. Indeed, suppose that a matrix A has a row a with a large Euclidean norm which is orthogonal to all the other rows. Also assume that the top eigenvector of the matrix A is in this direction. Since, the matrices $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_t$ are non-overlapping substreams of the matrix A, at most one of the matrices $\boldsymbol{B}_j$ can have the row a and hence the vector $v'/\|v'\|_2$ will not be a good approximation to $a/\|a\|_2$, the top eigenvector. Thus, we need to handle the heavy rows separately. We show that, by storing all the rows with a Euclidean norm larger than $\|A\|_{\mathrm{F}}/\sqrt{d \cdot \mathrm{polylog}(d)}$ and running the above described algorithm on the remaining set of rows, we can obtain a good approximation to the top eigenvector.

Our lower bound (Theorem 12.1.2) shows that any single-pass streaming algorithm must use space proportional to the number of heavy rows, and therefore our procedure that handles the heavy rows separately gives near-optimal bounds.

Finally, the row norm sampling technique of [MI10] serves as a general technique to reduce the number of rows in the stream while (approximately) preserving the top eigenvector. We use this observation to improve the $R = \Omega(\log n \cdot \log d)$ for arbitrary streams in [Pri23] to $R = \Omega(\log^2 d)$. We then show that assuming a uniformly random order, the analysis of [Pri23] can be improved to show that $R = \Omega(\log d)$ suffices. Thus, for random order streams, techniques before our work can be used to approximate the top eigenvector when the gap $R = \Omega(\log d)$. Our work improves upon this to give an algorithm that works for streams with $R = \Omega(1)$.

Organization. We first introduce the row-norm sampling procedure to obtain approximate quadratic forms. The proof is a slight modification of that of [MI10]. The only difference is that we instead consider a version that samples each row in the input independently with some appropriate probability

[3]Note that $A^{\mathrm{T}}A \cdot v = \sum_i \langle a_i, v\rangle a_i$.
[4]We use bold symbols to denote random variables.

and keeps the rows that are sampled after scaling appropriately. We then introduce and analyze our block power iteration algorithm when all rows have same Euclidean norms, and then extend it to the general case, which is our main result. Finally, we provide a lower bound showing that $\Omega(td/R)$ bits of space is necessary to obtain constant correlation with the top eigenvector.

12.2 Power Method with Approximate Quadratic Forms

In this section, we present and analyze our algorithm for approximating the top eigenvector of $A^{\mathrm{T}}A$ when the rows of A are presented to the algorithm in a uniformly random order.

We first show a row sampling technique that reduces the number of rows in the stream. The row-norm sampling technique for approximating the quadratic form $A^{\mathrm{T}}A$ with spectral norm guarantees was given by [MI10]. The technique works irrespective of the order of the rows.

12.2.1 Sampling for Row Reduction

Theorem 12.2.1. *Let A be an arbitrary $n \times d$ matrix. Given $p \in [0,1]^n$, let $\boldsymbol{Q}$ be an $n \times n$ diagonal matrix such that for each $i \in [n]$, we independently set $\boldsymbol{Q}_{ii} = 1/\sqrt{p_i}$ with probability p_i and 0 otherwise. If for all i,*

$$p_i \geq \min\left(1, C\frac{\|a_i\|_2^2}{\varepsilon^2\|A\|_2^2}\log d\right),$$

then with probability $1 - 1/\mathrm{poly}(d)$, $\|A^{\mathrm{T}}A - A^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}A\|_2 \leq \varepsilon\|A\|_2^2$. With probability at least $1 - 1/\mathrm{poly}(d)$, the matrix $\boldsymbol{Q}$ has at most $O(\varepsilon^{-2}\rho\log d)$ non-zero entries, where $\rho = \|A\|_{\mathrm{F}}^2/\|A\|_2^2$ denotes the stable rank of matrix A.

Proof. Let $\boldsymbol{X}_i$ denote an indicator random variable which denotes if $\boldsymbol{Q}_{ii}$ is nonzero. Note $\mathbf{E}[\boldsymbol{X}_i] = p_i$ and $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ are independent. Define a $d \times d$ random matrix $\boldsymbol{Y}_i = (\boldsymbol{X}_i/p_i - 1)a_ia_i^{\mathrm{T}}$, where a_i denotes the i-th row of A. We note that

$$A^{\mathrm{T}}A - A^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}A = \sum_{i=1}^{n}(\boldsymbol{X}_i/p_i - 1)a_ia_i^{\mathrm{T}} = \sum_{i=1}^{n}\boldsymbol{Y}_i.$$

We use the Matrix Bernstein inequality [Tro15] to bound $\|\sum_i \boldsymbol{Y}_i\|_2$. We first uniformly upper bound $\|\boldsymbol{Y}_i\|_2$. If $p_i = 1$, by definition $\|\boldsymbol{Y}_i\|_2 = 0$ with probability 1. Let $p_i \neq 0$. Then, $\|(\boldsymbol{X}_i/p_i - 1)a_ia_i^{\mathrm{T}}\|_2 \leq \|a_ia_i^{\mathrm{T}}\|_2/p_i \leq \varepsilon^2\|A\|_2^2/C\log d$ with probability 1.

We now bound $\| \sum_i \mathbf{E}[\boldsymbol{Y}_i^2] \|_2$.

$$\begin{aligned}
\sum_i \mathbf{E}[\boldsymbol{Y}_i^2] &= \sum_i \mathbf{E}[(1/p_i - 1)^2] \|a_i\|_2^2 a_i a_i^{\mathrm{T}} \\
&= \sum_{i:p_i>0} (1/p_i - 1) \|a_i\|_2^2 a_i a_i^{\mathrm{T}} \\
&\preceq \sum_{i:p_i>0} \frac{\varepsilon^2 \|A\|_2^2}{C\|a_i\|_2^2 \log d} \|a_i\|_2^2 a_i a_i^{\mathrm{T}} \\
&\preceq \frac{\varepsilon^2 \|A\|_2^2}{C \log d} A^{\mathrm{T}} A
\end{aligned}$$

which implies $\| \sum_i \mathbf{E}[\boldsymbol{Y}_i^2] \|_2 \le \varepsilon^2 \|A\|_2^4 / (C \log d)$. Now, we obtain

$$\begin{aligned}
\mathbf{Pr}[\| \sum_i \boldsymbol{Y}_i \|_2 \ge \varepsilon \|A\|_2^2] &\le 2d \cdot \exp\left(-\frac{\varepsilon^2 \|A\|_2^4 / 2}{\varepsilon^2 \|A\|_2^4/(C \log d) + \varepsilon^3 \|A\|_2^4 / (3C \log d)} \right) \\
&\le 2d \cdot \exp\left(-\frac{C \log d}{2(1 + \varepsilon/3)} \right).
\end{aligned}$$

If $C \ge 6(1 + \varepsilon/3)$, then $\mathbf{Pr}[\| \sum_i \boldsymbol{Y}_i \|_2 \ge \varepsilon \|A\|_2^2] \le 1 - 2/d^2$ which implies that with probability $\ge 1 - 2/d^2$, $\|A^{\mathrm{T}} A - A^{\mathrm{T}} \boldsymbol{Q}^{\mathrm{T}} \boldsymbol{Q} A\|_2 \le \varepsilon \|A\|_2^2$.

Now, the number of non-zero entries in the matrix $\boldsymbol{Q}$ is equal to $\sum_i \boldsymbol{X}_i$. We note $\mathbf{E}[\sum_i \boldsymbol{X}_i] \le C\varepsilon^{-2} \rho \cdot \log d$. By a Chernoff bound, we obtain that $\sum_i \boldsymbol{X}_i = O(\varepsilon^{-2} \rho \cdot \log d)$ with probability $\ge 1 - 1/\mathrm{poly}(d)$. □

Note that given the value of $\|A\|_2$, the sampling procedure in this theorem can be performed in a stream. Additionally, as the original stream is uniformly randomly ordered, the sub-sampled stream is also uniformly randomly ordered assuming that the sampling is independent of the order of the rows.

Given that all the non-zero entries of the matrix have absolute value at least $1/\mathrm{poly}(nd)$ and at most $\mathrm{poly}(nd)$, we have that $\|A\|_2^2$ lies in the interval $[1/\mathrm{poly}(nd), \mathrm{poly}(nd)]$. Thus, we can guess the value of $\|A\|_2^2$ as $2^i/\mathrm{poly}(nd)$ for $i = 0, \ldots, O(\log(nd))$ and one of these values must be a 2-approximation for $\|A\|_2^2$, and thus sub-sampling the rows using that guess satisfies the conditions in the above theorem. We can run the streaming algorithms on all the streams simultaneously to obtain $O(\log nd)$ vectors $u_1, \ldots, u_{O(\log nd)}$ as the candidates for being an approximation to the top eigenvector. From Theorem 12.2.1, the candidate vector u_j computed on the stream obtained by sampling the rows with the correct probabilities is a good approximation to the top eigenvector, and therefore $\|A \cdot u_j\|_2$ is large for that value of j. Thus, the vector u_j with the largest value $\|A \cdot u_j\|_2$ is a good approximation for the top eigenvector v_1. If $\boldsymbol{G}$ is a Gaussian matrix with $O(\varepsilon^{-2} \log d)$ rows, then for all u_j, we can approximate $\|A \cdot u_j\|_2$ up to a $1 \pm \varepsilon$ factor using $\|\boldsymbol{G} \cdot A \cdot u_j\|_2$ using the

Johnson-Lindenstrauss lemma. Additionally, the matrix $\boldsymbol{G} \cdot A$ can be maintained in the stream using $O(\varepsilon^{-2} \cdot d \log d)$ bits (when we see a row a_i, we sample an independent Gaussian vector $\boldsymbol{g}_i$ and add $\boldsymbol{g}_i a_i^{\mathrm{T}}$ to an accumulator to maintain $\boldsymbol{G} \cdot A$). Thus, at the end of processing the stream, we can compute a vector u_j that has a large value $\|A \cdot u_j\|_2$, and hence is a good approximation for v_1.

If we can process each created stream using s bits of space, then the overall space requirement is $O(s \cdot \log(nd) + d \cdot \operatorname{polylog}(d))$ bits, using $O(s)$ bits for each guess for the value of $\|A\|_2^2$ and $O(d \cdot \operatorname{polylog}(d))$ bits for storing a Gaussian sketch of the matrix with $\varepsilon = 1/\operatorname{polylog}(d)$.

12.2.2 Random-Order Streams with bounds on Norms

Algorithm 12.1: Approximate Eigenvector for Streams with no Large Norms

```
Input: An $n \times d$ matrix $A$ with $n = \Omega(\eta \cdot \rho(A) \cdot \log^2 d/\varepsilon^2)$, $\max_i \|a_i\|_2^2 / \min_i \|a_i\|_2^2 \le \eta$
Output: A vector $\boldsymbol{z}$
1  $t \leftarrow \lceil C_1 \log d \rceil$
2  Compute $\boldsymbol{G} \cdot A$ in the stream where $\boldsymbol{G}$ is a Gaussian matrix with $O(\varepsilon^{-2} \log d)$ rows
3  for $\rho = 1, 2, 4, \ldots, d$ simultaneously do
4      $p \leftarrow C_2 \eta \rho \log d / n\varepsilon^2$                // $p \le 1/(5t)$ for $\rho \le 2 \cdot \rho(A)$
5      $\boldsymbol{z}_\rho \sim N(0,1)^d$
6      for $j = 1, \ldots, t$ do
7          $\boldsymbol{y}_j \leftarrow \mathrm{Bin}(n, p)$
8          if $\boldsymbol{y}_j > 2np$ then
9              return $\perp$
10         end
           // The matrix $A_{j \cdot (2np) : j \cdot (2np) + \boldsymbol{y}_j}$ corresponds to $\boldsymbol{B}_j$ in the analysis.
11         $acc \leftarrow 0$
12         for $i = (j-1) \cdot (2np) + 1, \ldots, (j-1) \cdot (2np) + \boldsymbol{y}_j$ do
13             $acc \leftarrow acc + \langle a_i, \boldsymbol{z}_\rho \rangle \cdot a_i$
14         end
           // Here $acc = \boldsymbol{B}_j^{\mathrm{T}} \boldsymbol{B}_j \boldsymbol{z}_\rho$
15         $\boldsymbol{z}_\rho \leftarrow acc$
16         $\boldsymbol{z}_\rho \leftarrow \boldsymbol{z}_\rho / \|\boldsymbol{z}_\rho\|_2$
17     end
18 end
19 return $\arg\max_{z \in \{z_1, z_2, z_4, \ldots, z_d\}} \|(\boldsymbol{G} \cdot A)\boldsymbol{z}\|_2$
```

We now present the analysis of the block power method for random order streams assuming that the Euclidean norms of all the rows in A are close to each other. We later remove this assumption. Suppose there exists a parameter η such that $(\max_i \|a_i\|_2^2)/(\min_i \|a_i\|_2^2) \le \eta$. If η is close to 1 then all the rows in the stream have roughly the same norm.

Let $p = C\eta\rho\log(d)/\varepsilon^2 n$. We can see that for any row a_i in the stream,

$$C\frac{\|a_i\|_2^2}{\varepsilon^2\|A\|_2^2}\log d \le C\frac{\eta\|A\|_{\mathrm{F}}^2/n}{\varepsilon^2\|A\|_2^2}\log d \le \frac{C\eta\rho\log d}{n\varepsilon^2} = p.$$

Thus, p is greater than the probability with which we need to sample each row in the row-norm sampling result in Theorem 12.2.1. Now if we perform such a sampling of the rows of A, we sample $\mathrm{Bin}(n,p)$[5] number of rows, which is tightly concentrated around $np = \varepsilon^{-2}C\eta\rho\log d$. Thus, if we first sample $\boldsymbol{y} \sim \mathrm{Bin}(n,p)$ and then consider the first $\boldsymbol{y}$ number of rows in the random order stream, then we will have sampled from a distribution satisfying the requirements in Theorem 12.2.1 and can therefore obtain a matrix $\boldsymbol{B}$ such that

$$\|\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} - A^{\mathrm{T}}A\|_2 \le \varepsilon\|A\|_2^2.$$

Thus, assuming that the rows appear in a uniformly random order lets us show that the first $\boldsymbol{y}$ rows of the stream can be used to compute an approximation to the quadratic form $A^{\mathrm{T}}A$. We will now show that we can obtain $O(\log d)$ such quadratic forms in the stream given that the stream is long enough.

Assume that the number of rows in the stream $n = \Omega(\eta\rho\log^2 d/\varepsilon^2)$. We partition the stream into $t = \Theta(\log d)$ groups as follows: the first $2np$ rows are placed in the group 1, the second $2np$ rows are placed in the group 2, and so on. Note that since $n = \Omega(\eta\rho\log^2 d/\varepsilon^2)$, we can form t such groups. Since the rows are uniformly randomly ordered, the joint distribution of the rows appearing in group 1 is the same as that of the joint distribution of the rows appearing in group 2 and so on. Let $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t \sim \mathrm{Bin}(n,p)$ be drawn independently. With probability $\ge 1 - 1/\mathrm{poly}(d)$, we have $\boldsymbol{y}_i \le (3/2)np$ for all i. For $i = 1, \ldots, t$, let $\boldsymbol{B}_i$ be the matrix formed by the first $\boldsymbol{y}_i$ rows in group i. Using a union bound, we have that with probability $\ge 1 - 1/\mathrm{poly}(d)$, for all $i = 1, \ldots, t$,

$$\|A^{\mathrm{T}}A - \frac{1}{p}\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{B}_i\|_2 \le \varepsilon\|A\|_2^2.$$

Conditioned on the above event, we will now show that running the power method on the blocks $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_t$ lets us approximate the top singular vector of the matrix A.

Assumption 12.2.2. We assume that $\sigma_1(A)/\sigma_2(A) \ge 2$.

Lemma 12.2.3. *Let $\varepsilon > 1/\mathrm{poly}(d)$ be an accuracy parameter and $t = \Omega(\log d)$ be the number of iterations. Let $\varepsilon \le c/t^2$ for a small constant c. Suppose $B_1, \ldots, B_t$ all satisfy $\|A^{\mathrm{T}}A - B_j^{\mathrm{T}}B_j\|_2 \le \varepsilon\|A\|_2^2$ for $\varepsilon < 1/5$.*

[5]$\mathrm{Bin}(n,p)$ denotes the binomial distribution with parameters n and p.

If $\boldsymbol{g}$ is a random vector sampled from the Gaussian distribution, then the unit vector

$$\hat{v} \coloneqq \frac{(B_t^{\mathrm{T}} B_t) \cdots (B_1^{\mathrm{T}} B_1)\boldsymbol{g}}{\|(B_t^{\mathrm{T}} B_t) \cdots (B_1^{\mathrm{T}} B_1)\boldsymbol{g}\|_2}$$

satisfies

$$\langle \hat{v}, v_1 \rangle^2 \geq \frac{1}{1 + C' t\sqrt{\varepsilon}}$$

with probability $\geq 9/10$ for a large enough constant C'. Here v_1 denotes the top right singular vector of the matrix A.

Proof. Define $M \coloneqq (B_t^{\mathrm{T}} B_t) \cdots (B_1^{\mathrm{T}} B_1)$. Our strategy is to show that if v_1 is the top singular vector of the matrix A, then $\|v_1^{\mathrm{T}} M\|_2$ is comparable to $\|M\|_{\mathrm{F}}$ given that $\sigma_1(A)/\sigma_2(A) \geq 2$. We can then prove the lemma using simple properties of the Gaussian vector $\boldsymbol{g}$.

For an arbitrary j, let $(B_j^{\mathrm{T}} B_j)v_1 = \alpha v_1 + \Delta$ where $\Delta \perp v_1$. We note that $v_1^{\mathrm{T}}(B_j^{\mathrm{T}} B_j)v_1 = \alpha$. We have $\alpha = v_1^{\mathrm{T}} B_j^{\mathrm{T}} B_j v_1 \geq (1-\varepsilon)\sigma_1(A)^2$ using the fact that $\|B_j^{\mathrm{T}} B_j - A^{\mathrm{T}} A\|_2 \leq \varepsilon \|A\|_2^2$ and $v_1^{\mathrm{T}} A^{\mathrm{T}} A v_1 = \sigma_1(A)^2 = \|A\|_2^2$. If we show that Δ is small, then the vector $(B_j^{\mathrm{T}} B_j)v_1$ is oriented in a direction very close to that of v_1. Note that

$$\|(B_j^{\mathrm{T}} B_j)v_1\|_2 \leq \|B_j^{\mathrm{T}} B_j\|_2 \leq (1+\varepsilon)\sigma_1(A)^2$$

and $\|(B_j^{\mathrm{T}} B_j)v_1\|_2^2 = \alpha^2 + \|\Delta\|_2^2$ which implies $\|\Delta\|_2^2 \leq ((1+\varepsilon)^2 - (1-\varepsilon)^2)\sigma_1(A)^4 = 4\varepsilon \cdot \sigma_1(A)^4$ and thus $\|\Delta\|_2 \leq \sqrt{4\varepsilon}\sigma_1(A)^2$. Now,

$$\begin{aligned}
&\|M^{\mathrm{T}} v_1\|_2 \\
&= \|(B_1^{\mathrm{T}} B_1) \cdots (B_{t-1}^{\mathrm{T}} B_{t-1})(\langle B_t^{\mathrm{T}} B_t v_1, v_1 \rangle v_1 + \Delta_1)\|_2 \\
&\geq \langle B_t^{\mathrm{T}} B_t v_1, v_1 \rangle \|(B_1^{\mathrm{T}} B_1) \cdots (B_{t-1}^{\mathrm{T}} B_{t-1})v_1\|_2 - \|(B_1^{\mathrm{T}} B_1) \cdots (B_{t-1}^{\mathrm{T}} B_{t-1})\|_2 \|\Delta_1\|_2 \\
&\geq ((1-\varepsilon)\sigma_1(A)^2)\|(B_1^{\mathrm{T}} B_1) \cdots (B_{t-1}^{\mathrm{T}} B_{t-1})v_1\|_2 - (\sqrt{4\varepsilon}\sigma_1(A)^2)\|(B_1^{\mathrm{T}} B_1) \cdots (B_{t-1}^{\mathrm{T}} B_{t-1})\|_2.
\end{aligned}$$

Expanding similarly, we obtain

$$\|M^{\mathrm{T}} v_1\|_2 \geq (1-\varepsilon)^t \sigma_1(A)^{2t} - t\sqrt{4\varepsilon}(1+\varepsilon)^{t-1}\sigma_1(A)^{2t}.$$

Assuming $\varepsilon \leq c/t$ for a small constant c, we note that $(1-\varepsilon)^t \geq (1-2t\varepsilon)$ and $(1+\varepsilon)^t \leq (1+2t\varepsilon)$ which implies

$$\|M^{\mathrm{T}} v_1\|_2 = \|(B_1^{\mathrm{T}} B_1) \cdots (B_t^{\mathrm{T}} B_t)v_1\|_2 \geq (1 - 2t\varepsilon - 4t\sqrt{\varepsilon})\sigma_1(A)^{2t}.$$

We shall now show a bound on $\|M\|_{\mathrm{F}} = \|(B_t^{\mathrm{T}} B_t) \cdots (B_1^{\mathrm{T}} B_1)\|_{\mathrm{F}}$ which lets us show that the unit vector $\hat{v}$ is highly correlated with v_1. To bound the quantity $\|M\|_{\mathrm{F}}$, we first note the following facts:

1. $\|B_j^{\mathrm{T}} B_j\|_2 \le (1+\varepsilon)\sigma_1(A)^2$, and
2. $\sigma_2(B_j^{\mathrm{T}} B_j) \le \sigma_2(A)^2 + \varepsilon\sigma_1(A)^2 \le (1/4+\varepsilon)\sigma_1(A)^2$ by our gap assumption.

Now, we use the following theorem.

Theorem 12.2.4 ([WX97, Theorem 3(ii)]). *For any $r > 0$ and any matrices $A_1, \ldots, A_t$,*

$$\sum_i (\sigma_i(A_1 \cdots A_t))^r \le \sum_i \sigma_i(A_1)^r \cdots \sigma_i(A_t)^r.$$

Applying the above theorem with $r = 2$, we obtain

$$\begin{aligned}\|(B_t^{\mathrm{T}} B_t)\cdots(B_1^{\mathrm{T}} B_1)\|_{\mathrm{F}}^2 &\le (1+\varepsilon)^{2t}\sigma_1(A)^{4t} + (d-1)(1/4+\varepsilon)^t\sigma_1(A)^{4t}\\ &\le (1+4t\varepsilon)\sigma_1(A)^{4t} + \frac{d}{3^t}\sigma_1(A)^{4t}.\end{aligned}$$

When $t \ge 3\log(d/\varepsilon)$, we have $\|(B_t^{\mathrm{T}} B_t)\cdots(B_1^{\mathrm{T}} B_1)\|_{\mathrm{F}}^2 \le (1+4t\varepsilon+\varepsilon)\sigma_1(A)^{4t}$. We now use the following lemma.

Lemma 12.2.5. *Let $\boldsymbol{g}$ be a Gaussian random vector with each of the components being an independent standard Gaussian random variable. Let $\hat{v} = M\boldsymbol{g}/\|M\boldsymbol{g}\|_2$. For any unit vector v, with probability $\ge 4/5$,*

$$|\langle \hat{v}, v\rangle|^2 \ge \frac{1}{1 + C\frac{\|M\|_{\mathrm{F}}^2 - \|M^{\mathrm{T}}v\|_2^2}{\|M^{\mathrm{T}}v\|_2^2}}$$

for a large enough universal constant C.

Proof. Since v is a unit vector, we can write $\|M\boldsymbol{g}\|_2^2 = |v^{\mathrm{T}}M\boldsymbol{g}|^2 + \|(I - vv^{\mathrm{T}})M\boldsymbol{g}\|_2^2$. Hence, we have

$$|\langle \hat{v}, v\rangle|^2 = \frac{|v^{\mathrm{T}}Mg|^2}{\|M\boldsymbol{g}\|_2^2} = \frac{1}{1 + \frac{\|(I-vv^{\mathrm{T}})M\boldsymbol{g}\|_2^2}{|v^{\mathrm{T}}M\boldsymbol{g}|^2}}.$$

We now note that $v^{\mathrm{T}}M\boldsymbol{g} \sim N(0, \|M^{\mathrm{T}}v\|_2^2)$ and $\mathbf{E}[\|(I-vv^{\mathrm{T}})M\boldsymbol{g}\|_2^2] = \mathrm{tr}(M^{\mathrm{T}}(I-vv^{\mathrm{T}})M) = \|M\|_{\mathrm{F}}^2 - \|M^{\mathrm{T}}v\|_2^2$. By a union bound, with probability $\ge 4/5$, we have

$$\frac{\|(I - vv^{\mathrm{T}})M\boldsymbol{g}\|_2^2}{|v^{\mathrm{T}}M\boldsymbol{g}|^2} \le C\frac{\|M\|_{\mathrm{F}}^2 - \|M^{\mathrm{T}}v\|_2^2}{\|M^{\mathrm{T}}v\|_2^2}$$

for a large enough constant C. Therefore, with probability $\geq 4/5$, we get that

$$|\langle \hat{v}, v\rangle|^2 \geq \frac{1}{1 + C\frac{\|M\|_{\mathrm{F}}^2 - \|M^{\mathrm{T}}v\|_2^2}{\|M^{\mathrm{T}}v\|_2^2}}. \qquad \square$$

Applying the above lemma for $M = (B_t^{\mathrm{T}}B_t)\cdots(B_1^{\mathrm{T}}B_1)$ and $v = v_1$, we obtain

$$|\langle \hat{v}, v_1\rangle|^2 \geq \frac{1}{1 + C' t\sqrt{\varepsilon}}$$

with probability $\geq 4/5$. $\square$

If $t = \Theta(\log d)$ and $1/\mathrm{poly}(d) \leq \varepsilon \leq c/(\log d)^2$, then the above lemma shows that $\hat{v}$ has a large correlation with the top singular vector v_1. Using this lemma, we show that Algorithm 12.1 can be used to obtain an approximation for v_1 in random order streams with bounded norms.

Theorem 12.2.6. *Let $\alpha \geq 1/\mathrm{poly}(d)$ be an accuracy parameter. Let η be a parameter such that $\frac{\max_i \|a_i\|_2^2}{\min_i \|a_i\|_2^2} \leq \eta$. If the number of rows in the stream $n = \Omega(\alpha^{-4} \cdot \rho(A) \cdot \eta \cdot \log^6 d)$, where $\rho(A) = \|A\|_{\mathrm{F}}^2/\|A\|_2^2$ and the rows in the stream are ordered uniformly at random, then we can compute a vector $\hat{v}$ using the block power method that satisfies*

$$|\langle v_1, \hat{v}\rangle|^2 \geq 1 - 3\alpha$$

with probability $\geq 4/5$ if $\sigma_1(A)/\sigma_2(A) \geq 2$. The algorithm uses $O(d \cdot \mathrm{polylog}(d)/\alpha^4)$ bits of space.

Proof. Set $\varepsilon = \alpha^2/C\log^2 d$ for a large enough constant C. Assuming $n = \Omega(\alpha^{-4}\rho\eta\log^6 d)$, we have $n = \Omega(\varepsilon^{-2}\rho\eta\log^2 d)$. Now consider the execution of Algorithm 12.1 on matrix A, with parameters η and ε. Let $\rho = 2^j$ be such that $\rho(A)/2 \leq \rho \leq \rho(A)$, and consider the execution in the algorithm with parameter ρ. Using Theorem 12.2.1, with probability $\geq 1 - 1/\mathrm{poly}(d)$, the algorithm computes t matrices $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_t$ such that for all $j \in [t]$,

$$\|\frac{1}{p}\boldsymbol{B}_j^{\mathrm{T}}\boldsymbol{B}_j - A^{\mathrm{T}}A\|_2 \leq \varepsilon\|A\|_2^2.$$

Noting that $\boldsymbol{z}_\rho = (\boldsymbol{B}_t^{\mathrm{T}}\boldsymbol{B}_t)\cdots(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)\boldsymbol{g}/\|(\boldsymbol{B}_t^{\mathrm{T}}\boldsymbol{B}_t)\cdots(\boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{B}_1)\boldsymbol{g}\|_2$, by Lemma 12.2.3, we have with probability $\geq 9/10$ that

$$\langle \boldsymbol{z}_\rho, v_1\rangle^2 \geq \frac{1}{1 + C't\sqrt{\varepsilon}} \geq 1 - \alpha.$$

Thus, for ρ which satisfies $\rho(A)/2 \leq \rho \leq \rho(A)$, the algorithm computes a vector $\boldsymbol{z}_\rho$ that has a large correlation with the vector v_1. Since the algorithm does not know the exact value of ρ, it computes an approximation for $\|A\boldsymbol{z}\|_2^2$ for all $\boldsymbol{z} \in \{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_4, \ldots, \boldsymbol{z}_d\}$. First, we condition on the fact that with probability $\geq 1 - 1/\mathrm{poly}(d)$, for all z_i, $\|GAz_i\|_2^2 = (1 \pm \varepsilon)\|Az_i\|_2^2$. Since $\langle z_\rho, v_1 \rangle^2 \geq (1 - \alpha)$, we note that $\|GAz_\rho\|_2^2 \geq (1 - \varepsilon)(1 - \alpha)\sigma_1(A)^2$. Now, for the vector z returned by the algorithm, we have $\|Az\|_2^2 \geq (1 - O(\varepsilon))(1 - \alpha)\sigma_1(A)^2$ which implies that

$$\langle z, v_1 \rangle^2 \cdot \sigma_1(A)^2 + (1 - \langle z, v_1 \rangle^2)\frac{\sigma_1(A)^2}{R} \geq \|Az\|_2^2 \geq (1 - \alpha - O(\varepsilon))\sigma_1(A)^2$$

and therefore $\langle z, v_1 \rangle^2 \geq 1 - 3\alpha$ since $R \geq 2$. □

12.2.3 Random Order Streams without Norm Bounds

Assuming that the random order streams are long enough, Theorem 12.2.6 shows that if all the squared row norms are within an η factor, then the block power method outputs a vector with a large correlation with the top eigenvector of the matrix $A^{\mathrm{T}}A$. For general streams, the factor η could be quite large and hence the algorithm requires very long streams to output an approximation to v_1.

If there are no *heavy* rows, i.e., rows with a Euclidean norm larger than $\|A\|_{\mathrm{F}}/\sqrt{d \cdot \mathrm{polylog}(d)}$, then the row norm sampling procedure in Theorem 12.2.1 can be used to convert any randomly ordered stream of rows into a uniformly random stream of rows that all have the same norm. The row norm sampling procedure computes a probability $p_i = \min(1, C\varepsilon^{-2}\|a_i\|_2^2 \log d/\|A\|_2^2)$ and samples the row a_i with probability p_i. If sampled, then the row a_i is scaled by $1/\sqrt{p_i}$. From Theorem 12.2.1, we have that the top eigenvector of the *quadratic form* of the sampled-and-rescaled submatrix is a good approximation to the top eigenvector $A^{\mathrm{T}}A$ when the gap R is large enough. Suppose $p_i < 1$. If the row a_i is sampled, we then have

$$\|a_i/\sqrt{p_i}\|_2 = \frac{\varepsilon\|A\|_2}{\sqrt{C \log d}}.$$

Thus, if $p_i < 1$ for all i, then all the sampled-and-rescaled rows have the same Euclidean norm and therefore, we can run the algorithm from Theorem 12.2.6 by setting $\eta = 1$. Note that $p_i = 1$ only if $\|a_i\|_2^2 \geq \varepsilon^2\|A\|_2^2/C\log(d)$. Since we assumed that there are no heavy rows, there is no row with $p_i = 1$ as long as $\varepsilon \geq 1/\mathrm{polylog}(d)$. Thus, using Theorem 12.2.6 on the row norm sampled sub stream directly gives us a good approximation to the top eigenvector. However, in general, the streams can have rows with large Euclidean norm. We will now state our theorem and describe how such streams can be handled.

Theorem 12.2.7. *Let A be an $n \times d$ matrix with its non-zero entries satisfying $1/\mathrm{poly}(d) \leq |A_{i,j}| \leq \mathrm{poly}(d)$, and hence representable using $O(\log d)$ bits of precision. Let $R = \sigma_1(A)^2/\sigma_2(A)^2$. Assume $2 \leq R \leq C_1 \log^2 d$. Let h be the number of rows in A with norm at most $\|A\|_{\mathrm{F}}/\sqrt{d \cdot \mathrm{polylog}(d)}$, where we use $\mathrm{polylog}(d) = \log^{C_2} d$ for a large enough universal constant C_2. Given the rows of the matrix A in a uniformly random order, there is an algorithm using $O((h+1) \cdot d \cdot \mathrm{polylog}(d) \cdot \log n)$ bits of space and which outputs a vector $\hat{v}$ such that with probability $\geq 4/5$, $\hat{v}$ satisfies $\langle \hat{v}, v_1 \rangle^2 \geq 1 - 8/\sqrt{R}$, where v_1 is the top eigenvector* *of the matrix $A^{\mathrm{T}}A$.*

Proof. Partition the matrix A into A_{light} and A_{heavy}, where A_{heavy} is the submatrix with rows a_i such that $\|a_i\|_2 > \|A\|_{\mathrm{F}}/\sqrt{d \cdot \operatorname{polylog}(d)}$ and A_{light} is the remaining rows. From our assumption, the number of rows in A_{heavy} is at most h. Note that given a uniformly random stream of rows of A, we can obtain a uniformly random stream of rows of A_{light} by just filtering out the rows in A_{heavy}.

Suppose, $\|A_{\text{heavy}} \cdot v_1\|_2 \geq (1-\beta)\|A\|_2$ for a parameter β to be chosen later. Let v_1' be the top singular vector of the matrix A_{heavy}. Note

$$\|A \cdot v_1'\|_2^2 \geq \|A_{\text{heavy}} \cdot v_1'\|_2^2 \geq \|A_{\text{heavy}} \cdot v_1\|_2^2 \geq (1-\beta)^2\|A\|_2^2,$$

and therefore we have $\langle v_1', v_1\rangle^2 \geq 1 - 4\beta$, assuming $R \geq 2$. Thus, while processing the stream, we can store all the heavy rows and at the end of the stream compute the top right singular vector of A_{heavy}, in order to obtain a good approximation for v_1.

Suppose $\|A_{\text{heavy}} \cdot v_1\|_2 \leq (1-\beta)\|A\|_2$. This implies $\|A_{\text{light}} \cdot v_1\|_2^2 \geq \|A\|_2^2 - \|A_{\text{heavy}} \cdot v_1\|_2^2 \geq \beta \cdot \|A\|_2^2$. If we set $\beta \geq 2/R$, we have

$$\frac{\sigma_1(A_{\text{light}})^2}{\sigma_2(A_{\text{light}})^2} \geq \frac{\beta\|A\|_2^2}{\sigma_2(A)^2} \geq 2.$$

Let v_1' be the top singular vector of A_{light}. We will describe how to approximate v_1'. Consider applying the row norm sampling procedure with parameter ε to the matrix A_{light}. Given a row $a_i \in A_{\text{light}}$ the corresponding sampling probability p_i is given by

$$p_i = \frac{C \log d \cdot \|a_i\|_2^2}{\varepsilon^2 \|A_{\text{light}}\|_2^2} \leq \frac{C \log d \cdot \|A\|_{\mathrm{F}}^2/(d \cdot \operatorname{polylog}(d))}{\varepsilon^2\beta^2\|A\|_2^2} \leq \frac{C}{\varepsilon^2\beta^2 \operatorname{polylog}(d)}.$$

Assuming that $\varepsilon^2\beta^2 \geq 1/\operatorname{polylog}(d)$, we obtain that $p_i < 1$ for all the rows in the matrix A_{light}. Let $\boldsymbol{B}_{\text{light}}$ be the matrix obtained after applying the row norm sampling procedure to the matrix A_{light}. Note that $\rho(\boldsymbol{B}_{\text{light}}) \approx \rho(\boldsymbol{A}_{\text{light}})$ and the number of rows in $\boldsymbol{B}_{\text{light}}$ is $\Theta(\rho(A_{\text{light}}) \cdot \log d \cdot \varepsilon^{-2})$, and therefore $\Theta(\rho(\boldsymbol{B}_{\text{light}}) \cdot \log d \cdot \varepsilon^{-2})$. Setting $\varepsilon = \alpha^2/\log^{5/2} d$, we obtain that the number of rows in the matrix $\boldsymbol{B}_{\text{light}}$ is $\Theta(\alpha^{-4} \cdot \rho(\boldsymbol{B}_{\text{light}}) \cdot \log^6 d)$ and thus assuming $\varepsilon^2\beta^2 = \alpha^4\beta^2/\log^5 d \geq 1/\operatorname{polylog}(d)$, we can use Theorem 12.2.6 to obtain a vector $\hat{v}$ satisfying

$$\langle \hat{v}, v_1'\rangle^2 \geq 1 - 3\alpha.$$

We will now show that v_1' has a large correlation with v_1 which then implies $\hat{v}$ has a large correlation with v_1. Since $\|A_{\text{light}}\|_2 \geq \|A\|_2 - \|A_{\text{heavy}}\|_2 \geq \beta\|A\|_2$, $\|A_{\text{light}}\|_2^2 = \|A_{\text{light}} \cdot v_1'\|_2^2 \geq \beta\|A\|_2^2$. Consider the following upper bound on $\|A_{\text{light}} \cdot v_1'\|_2^2$:

$$\begin{aligned}\|A_{\text{light}}\|_2^2 = \|A_{\text{light}} \cdot v_1'\|_2^2 &= \|A_{\text{light}} \cdot (\langle v_1', v_1\rangle \cdot v_1 + (I - v_1 v_1^{\mathrm{T}})v_1')\|_2^2 \\ &= \|\langle v_1, v_1'\rangle A_{\text{light}} \cdot v_1 + A_{\text{light}}(I - v_1 v_1^{\mathrm{T}})v_1'\|_2^2 \\ &\le (1+\theta) \cdot \langle v_1, v_1'\rangle^2 \cdot \|A_{\text{light}} \cdot v_1\|_2^2 + (1 + 1/\theta) \cdot \|A_{\text{light}}(I - v_1 v_1^{\mathrm{T}})v_1'\|_2^2\end{aligned}$$

for any $\theta > 0$. Using the fact that the rows of the matrix A_{light} are a subset of the rows of the matrix A and that $\|A(I - v_1 v_1^{\mathrm{T}})\|_2 = \sigma_2(A) = \sigma_1(A)/\sqrt{R}$, we have

$$\begin{aligned}\|A_{\text{light}}\|_2^2 &\le (1+\theta) \cdot \langle v_1, v_1'\rangle^2 \cdot \|A_{\text{light}}\|_2^2 + (1+1/\theta) \cdot \frac{\sigma_1^2}{R} \cdot (1 - \langle v_1, v_1'\rangle^2) \\ &= \langle v_1, v_1'\rangle^2((1+\theta) \cdot \|A_{\text{light}}\|_2^2 - (1+1/\theta)\sigma_1^2/R) + (1+1/\theta) \cdot \sigma_1^2/R\end{aligned}$$

which implies

$$\begin{aligned}\langle v_1, v_1'\rangle^2 \ge \frac{\|A_{\text{light}}\|_2^2 - (1+1/\theta) \cdot \sigma_1^2/R}{(1+\theta)\|A_{\text{light}}\|_2^2 - (1+1/\theta)\sigma_1^2/R} &= 1 - \frac{\theta \cdot \|A_{\text{light}}\|_2^2}{(1+\theta)\|A_{\text{light}}\|_2^2 - (1+1/\theta)\sigma_1^2/R} \\ &\ge 1 - \frac{\theta}{1 + \theta - (1+1/\theta)/R\beta}\end{aligned}$$

using the fact that $\|A_{\text{light}}\|_2^2 \ge \beta^2\sigma_1^2$. Now assuming $R\beta \ge 1$ and picking $\theta = 2/(R\beta - 1)$, we obtain

$$\langle v_1, v_1'\rangle^2 \ge 1 - \frac{4R\beta}{(1+R\beta)^2} \ge 1 - \frac{4}{R\beta}.$$

We therefore have

$$\langle \hat{v}, v_1\rangle^2 \ge 1 - \frac{4}{R\beta} - 4\alpha. \tag{12.1}$$

Setting $\beta = 1/\sqrt{R}$ and $\alpha = 1/\sqrt{R}$, we satisfy all the requirements assuming that $R \le \text{polylog}(d)$ and obtain a vector $\hat{v}$ satisfying $\langle \hat{v}, v_1\rangle^2 \ge 1 - 8/\sqrt{R}$. When $\|A_{\text{heavy}}\|_2 \ge (1-\beta)\|A\|_2$, we already have a vector $v' =$ top eigenvector of A_{heavy} that satisfies $\langle \hat{v}, v_1\rangle^2 \ge 1 - 4\beta \ge 1 - 4/\sqrt{R}$. Thus, in both the cases, we obtain a vector $\hat{v}$ satisfying $\langle \hat{v}, v_1\rangle^2 \ge 1 - O(1/\sqrt{R})$.

The procedure described requires knowing the approximate values of $\|A\|_{\mathrm{F}}, \|A_{\text{light}}\|_2$. Since, we assume that all the non-zero entries of the matrix have an absolute value at least $1/\text{poly}(d)$ and at most $\text{poly}(d)$, the values $\|A\|_{\mathrm{F}}, \|A_{\text{light}}\|_2$ lie in the interval $[1/\text{poly}(d), \text{poly}(nd)]$. Hence, using $O(\log nd)$ guesses each for $\|A\|_{\mathrm{F}}$ and $\|A_{\text{light}}\|_2$ and using a Gaussian sketch of A similar to that in Algorithm 12.1, we can obtain a vector satisfying the guarantees in the theorem. □

12.3 Lower Bounds

Our algorithm uses $\widetilde{O}(h \cdot d)$ space when the number of heavy rows in the stream is h. We want to argue that it is nearly tight. We show the following theorem.

Theorem 12.3.1. *Given a dimension d, let h and R be arbitrary with $R \le h \le d$ and $R^2 \cdot h = O(d)$. Consider an algorithm $\mathscr{A}$ with the following property:*

Given any fixed matrix $n \times d$ matrix A with $O(h)$ heavy rows and gap $\sigma_1(A)^2/\sigma_2(A)^2 \ge R$, in the form of a uniform random order stream, the algorithm $\mathscr{A}$ outputs a unit vector $\hat{v}$ such that, with probability $\ge 1 - (1/2)^{4R+4}$ over the randomness of the stream and the internal randomness of the algorithm,

$$|\langle \hat{v}, v_1 \rangle|^2 \ge 1 - c/R^2.$$

If c is a small enough constant, then the algorithm $\mathscr{A}$ must use $\Omega(h \cdot d/R)$ bits of space.

The theorem shows that a streaming algorithm must use $\Omega(hd/R)$ bits of space assuming that with high probability, it outputs a vector with a large enough correlation with the top eigenvector of $A^{\mathrm{T}}A$ when the rows are given in a random order stream.

Our proof uses the same lower bound instance as that of [Pri23]. The key difference from Price's proof is that our lower bound must hold against random order streams.

Proof. For each $i \in [h]$, let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_h$ be drawn independently and uniformly at random from $\{+1, -1\}^d$. Let $\boldsymbol{i} \sim [h]$ be drawn uniformly at random, and for an integer k to be chosen later, let $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k \in \mathbb{R}^d$ be vectors that share the first $(1-\gamma)d$ coordinates with the vector $\boldsymbol{x}_{\boldsymbol{i}}$. Each of the last $\gamma \cdot d$ elements of each of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ are sampled uniformly at random from the set $\{+1, -1\}$. Define $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{h+k}$ such that for $j \le h$, $\boldsymbol{z}_j = \boldsymbol{x}_j$ and for $j > h$, let $\boldsymbol{z}_j = \boldsymbol{y}_{j-h}$.

Now consider the stream $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{h+k}$. Price argues that when $k \ge 4R$, the gap of this stream is at least R with large probability over the randomness used in the construction of the stream. Let $\boldsymbol{\pi} : [h+k] \to [h+k]$ be a uniformly random permutation independent of $\boldsymbol{i}$. Consider the following event $\mathscr{E}$:

$$\boldsymbol{\pi}(\boldsymbol{i}) \le h/2 \text{ and } \boldsymbol{\pi}(h+1), \ldots, \boldsymbol{\pi}(h+k) > h/2.$$

We have that the probability of the event $\mathscr{E}$ is

$$\frac{h/2+k}{h+k} \cdot \frac{h/2+k-1}{h+k-1} \cdots \frac{h/2+1}{h+1} \cdot \frac{h/2}{h} \ge (1/2)^{k+1}.$$

Let $S_{\boldsymbol{i}}$ be the set of permutations π that satisfy the above event. Therefore, we have $\mathbf{Pr}_{\boldsymbol{\pi}}[\boldsymbol{\pi} \in S_{\boldsymbol{i}}] \ge (1/2)^{k+1}$. If the probability of failure, δ, of the algorithm $\mathscr{A}$ satisfies $\delta \le (1/2)^{k+4}$, we have that

$$\mathbf{Pr}_{\boldsymbol{\pi},\ \text{internal randomness}}[\mathscr{A} \text{ succeeds on } \boldsymbol{z}_{\boldsymbol{\pi}(1)}, \ldots, \boldsymbol{z}_{\boldsymbol{\pi}(h+k)} \mid \boldsymbol{\pi} \in S_{\boldsymbol{i}}] \ge \frac{3}{4}.$$

Let $\boldsymbol{s}_{\text{mid}}$ be the state of the algorithm after $h/2$ steps and $\boldsymbol{s}_{\text{fin}}$ be the final state of the algorithm. The randomness in $\boldsymbol{s}_{\text{fin}}$ is from the following sources: (i) randomness of the vectors $\boldsymbol{x}_1, \dots, \boldsymbol{x}_h$, (ii) the index $\boldsymbol{i} \in [h]$, (iii) the vectors $\boldsymbol{y}_1, \dots, \boldsymbol{y}_k$, (iv) the permutation $\boldsymbol{\pi}$, and (v) the internal randomness of the algorithm. From here on, condition on the event $\mathcal{E}$, i.e., that the permutation $\boldsymbol{\pi} \in S_{\boldsymbol{i}}$. We will not explicitly mention that all entropy and information terms in the proof are conditioned on $\mathcal{E}$. Since $\boldsymbol{\pi}(\boldsymbol{i}) \le h/2$, we have

$$\boldsymbol{s}_{\text{fin}} \text{ is conditionally independent of } x_{\boldsymbol{i}}[(1-\gamma)\cdot d + 1 : d] \text{ given } \boldsymbol{s}_{\text{mid}}.$$

Using the data processing inequality, we obtain that

$$I(\boldsymbol{s}_{\text{mid}}; \boldsymbol{x}_{\boldsymbol{i}}[(1-\gamma)\cdot d + 1 : d]) \ge I(\boldsymbol{s}_{\text{fin}}; \boldsymbol{x}_{\boldsymbol{i}}[(1-\gamma)\cdot d + 1 : d]).$$

When $h \le cd/R^2$, $k = 4R$, $\gamma = 1/4$ and $\varepsilon \le c/k^2$ for a small constant, we have as in the proof of Theorem 1.3 in [Pri23] that,

$$I(\boldsymbol{s}_{\text{fin}}; \boldsymbol{x}_{\boldsymbol{i}}[(1-\gamma)\cdot d + 1 : d]) \ge \Omega(d/R)$$

which now implies

$$I(\boldsymbol{s}_{\text{mid}}; \boldsymbol{x}_{\boldsymbol{i}}[(1-\gamma)\cdot d + 1 : d]) \ge \Omega(d/R).$$

Note that conditioned on the event $\mathcal{E}$, the distribution of $\boldsymbol{i}$ is uniform over $\{\, \boldsymbol{\pi}^{-1}(1), \dots, \boldsymbol{\pi}^{-1}(h/2) \,\}$. We now prove the following lemma:

Lemma 12.3.2. *Let $Y_1, \dots, Y_\ell$ be independent random variables. Let $\boldsymbol{i} \sim [\ell]$ be a uniform random variable independent of $\boldsymbol{X}$. We have*

$$I(\boldsymbol{X}\,;\, Y_1) + \cdots + I(\boldsymbol{X}\,;\, Y_\ell) \ge \ell \cdot (I(\boldsymbol{X}; Y_{\boldsymbol{i}}) - \log_2 \ell).$$

Proof. By definition, we have

$$I(\boldsymbol{X}\,; Y_{\boldsymbol{i}}) = H(Y_{\boldsymbol{i}}) - H(Y_{\boldsymbol{i}} \mid \boldsymbol{X}).$$

Now, we note that $H(Y_{\boldsymbol{i}}) \le H(Y_{\boldsymbol{i}}, \boldsymbol{i}) = H(\boldsymbol{i}) + H(Y_{\boldsymbol{i}} \mid \boldsymbol{i}) = \log_2 \ell + \frac{H(Y_1)+\cdots+H(Y_\ell)}{\ell}$. We now lower bound $H(Y_{\boldsymbol{i}} \mid \boldsymbol{X})$. Since conditioning always decreases entropy, we obtain

$$H(Y_{\boldsymbol{i}} \mid \boldsymbol{X}) \ge H(Y_{\boldsymbol{i}} \mid \boldsymbol{i}, \boldsymbol{X}).$$

As $\boldsymbol{X}$ is independent of $\boldsymbol{i}$, we have

$$H(Y_{\boldsymbol{i}} \mid X) \geq H(Y_{\boldsymbol{i}} \mid \boldsymbol{i}, X) = \frac{H(Y_1 \mid X) + \cdots + H(Y_\ell \mid X)}{\ell}$$

which then implies

$$\begin{aligned} I(\boldsymbol{X} ; Y_{\boldsymbol{i}}) &\leq H(\boldsymbol{i}) + \frac{H(Y_1) + \cdots + H(Y_\ell)}{\ell} - \frac{H(Y_1 \mid X) + \cdots + H(Y_\ell \mid X)}{\ell} \\ &\leq H(\boldsymbol{i}) + \frac{I(X ; Y_1) + \cdots + I(X ; Y_\ell)}{\ell}. \end{aligned}$$

Since $H(\boldsymbol{i}) = \log_2 \ell$, we have the proof. □

Using this lemma,

$$\begin{aligned} &I(\boldsymbol{s}_{\text{mid}}; \boldsymbol{x}_{\boldsymbol{\pi}^{-1}(1)}[(1-\gamma)\cdot d + 1 : d]) + \cdots + I(\boldsymbol{s}_{\text{mid}}; \boldsymbol{x}_{\boldsymbol{\pi}^{-1}(h/2)}[(1-\gamma)\cdot d + 1 : d]) \\ &= (h/2) \cdot I(\boldsymbol{s}_{\text{mid}}; \boldsymbol{x}_{\boldsymbol{i}}[(1-\gamma)\cdot d + 1 : d] - \log_2(h/2)) \\ &\geq \Omega(hd/R) - h \log_2 h. \end{aligned}$$

Lemma 12.3.3. *If X, Y are independent, then $I(Z ; (X, Y)) \geq I(Z ; X) + I(Z ; Y)$.*

Proof.

$$\begin{aligned} I(Z ; (X, Y)) &= H((X, Y)) - H((X, Y) \mid Z) \\ &= H(X) + H(Y) - H((X, Y) \mid Z). \end{aligned}$$

Now, we note that for any three random variables X, Y, Z, we have $H((X, Y) \mid Z) \leq H(X \mid Z) + H(Y \mid Z)$ which proves the lemma. □

Using the independence of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_h$ conditioned on the event $\mathscr{E}$, we obtain

$$I(\boldsymbol{s}_{\text{mid}}; (\boldsymbol{x}_{\boldsymbol{\pi}^{-1}(1)}[(1-\gamma)\cdot d + 1 : d], \ldots, \boldsymbol{x}_{\boldsymbol{\pi}^{-1}(h/2)}[(1-\gamma)\cdot d + 1 : d])) \geq \Omega(hd/R) - h \log_2 h$$

which then implies

$$H(\boldsymbol{s}_{\text{mid}}) \geq \Omega(hd/R)$$

using the fact that $R^2 \cdot h = O(d)$. Finally, we have $\max |\boldsymbol{s}_{\text{mid}}| \geq \Omega(hd/R)$. Here $|\boldsymbol{s}_{\text{mid}}|$ is the number of bits used in the representation of the state $\boldsymbol{s}_{\text{mid}}$. □

12.4 Improving the Gap Requirements in Price's Algorithm

12.4.1 Arbitrary Order Streams

As discussed in Section 12.2.1, we can guess an approximation of $\|A\|_2^2$ in powers of 2 and sample at most $O(d \log d/\varepsilon^2)$ rows in the stream to obtain a matrix $\boldsymbol{B}$, in the form of a stream, satisfying $\|\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} - A^{\mathrm{T}}A\|_2 \le \varepsilon\|A\|_2^2$, with a large probability. Using Weyl's inequalities, we obtain that

$$\sigma_2(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}) \le \sigma_2(A^{\mathrm{T}}A) + \varepsilon\|A\|_2^2 \quad \text{and} \quad \sigma_1(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}) \ge (1-\varepsilon)\sigma_1(A^{\mathrm{T}}A)$$

implying $R' = \sigma_1(\boldsymbol{B})^2/\sigma_2(\boldsymbol{B})^2 \ge (1-\varepsilon)/(1/R + \varepsilon)$. For $\varepsilon = 1/(2R) \le 1/2$, we note $R' \ge R/3$. Let $n' = O(R^2 \cdot d \log d)$ be the number of rows in the matrix $\boldsymbol{B}$ and note that $R' = \Omega(\log n' \cdot \log d)$ assuming $R = \Omega(\log^2 d)$. Hence, running Price's algorithm on the rows of the matrix $\boldsymbol{B}$, we compute a vector $\hat{v}$ for which

$$|\langle \hat{v}, v_1' \rangle|^2 \ge 1 - \frac{\log d}{CR'} - \frac{1}{\mathrm{poly}(d)}$$

with a large probability, where v_1' is the top eigenvector of the matrix $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}$. We now note that if v_1 denotes the top eigenvector of the matrix $A^{\mathrm{T}}A$, then $|\langle v_1, v_1' \rangle|^2 \ge 1 - O(1/R)$ which therefore implies that with a large probability,

$$|\langle \hat{v}, v_1 \rangle|^2 \ge 1 - \frac{\log d}{CR}.$$

Thus, sub-sampling the stream using row norm sampling and then running Price's algorithm, we obtain an algorithm for arbitrary order streams with a gap $R = \Omega(\log^2 d)$.

12.4.2 Random Order Streams

Lemma 3.5 in Price's proof can be tightened when the rows of the stream are uniformly randomly ordered. Specifically, we want to bound the following quantity:

$$\sum_{i=1}^{n} \langle a_i, P\hat{v}_{i-1} \rangle^2$$

where $P = I - v_1 v_1^{\mathrm{T}}$ denotes the projection away from the top eigenvector, and $\hat{v}_{i-1}$ is a function of $v_1, a_1, \ldots, a_{i-1}$. We have

$$\mathbf{E}[\langle a_i, P\hat{v}_{i-1} \rangle^2] = \mathbf{E}[\mathbf{E}[\langle a_i, P\hat{v}_{i-1} \rangle^2 \mid a_1, \ldots, a_{i-1}]].$$

Given that the first $i-1$ rows are $a_1, \ldots, a_{i-1}$, assuming uniform random order, we have

$$\begin{aligned}\mathbf{E}[\langle a_i, P\hat{v}_{i-1}\rangle^2 \mid a_1, \ldots, a_{i-1}] &= \frac{1}{n-i+1}\hat{v}_{i-1}^{\mathrm{T}} P(A^{\mathrm{T}}A - a_1 a_1^{\mathrm{T}} - \cdots - a_{i-1}a_{i-1}^{\mathrm{T}})P\hat{v}_{i-1} \\ &\leq \frac{\sigma_2(A)^2}{n-i+1}.\end{aligned}$$

Hence, $\mathbf{E}[\langle a_i, P\hat{v}_{i-1}\rangle^2] \leq \sigma_2(A)^2/(n-i+1)$ and $\mathbf{E}[\sum_{i=1}^n \langle a_i, P\hat{v}_{i-1}\rangle^2] \leq \sigma_2(A)^2(1+\log n)$. Price defines $\eta \cdot \sigma_2(A)^2$ as σ_2 and in that notation, we obtain $\eta \sum_{i=1}^n \langle a_i, P\hat{v}_{i-1}\rangle^2 \leq 10\sigma_2(1+\log n)$ with probability $\geq 9/10$ by Markov's inequality. In the proof of Lemma 3.6 in Price's manuscript, if $\sigma_1/\sigma_2 \geq 20(1+\log_2 n)$, we obtain $\log \|v_n\|_2 \gtrsim \sigma_1$. Now, $\sigma_1 \geq O(\log d)$ ensures that the Proof of Theorem 1.1 in Price's manuscript goes through.

Using the row-norm sampling analysis from the previous section, we can assume $n = \mathrm{poly}(d)$ and therefore a gap of $O(\log d)$ between the top two eigenvalues of $A^{\mathrm{T}}A$ is enough for Oja's algorithm to output a vector with a large correlation with the top eigenvector in random order streams.

12.5 Hard Instance for Oja's Algorithm

At a high level, Price's algorithm runs Oja's algorithm with different learning rates η and in the event that the norm of the output vector with each of the learning rates η is small, then the row with the largest norm is output. The algorithm is simple and can be implemented using an overall space of $O(d \cdot \mathrm{polylog}(d))$ bits.

The algorithm initializes $z_0 = \boldsymbol{g}$ where $\boldsymbol{g}$ is a random Gaussian vector. The algorithm streams through the rows $a_1, \ldots, a_n$ and performs the following operation

$$z_i \leftarrow z_{i-1} + \eta \cdot \langle z_{i-1}, a_i\rangle a_i.$$

The algorithm computes the smallest learning rate η when $\|z_n\|_2$ is large enough, and then outputs either $z_n/\|z_n\|_2$ or $\bar{a}/\|\bar{a}\|_2$ as an approximation to the eigenvector of the matrix $A^{\mathrm{T}}A$. Here $\bar{a}$ denotes the row in A with the largest Euclidean norm.

The following theorem shows that at gaps $\leq O(\log d/\log\log d)$, we cannot use Oja's algorithm with a fixed learning rate η to obtain constant correlation with the top eigenvector.

Theorem 12.5.1. *Given dimension d, a constant $c > 0$, a parameter M, for all $R = O_c(\log d/\log\log d)$ there is a stream of vectors $a_1, \ldots, a_n \in \mathbb{R}^d$ with $n = O(R+M)$ such that:*

1. *$\sigma_1(A)^2/\sigma_2(A)^2 \geq R/2$, and*
2. *Oja's algorithm with any learning rate $\eta < M$ fails to output a unit vector $\hat{v}$ that satisfies, with probability $\geq 9/10$,*

$$|\langle \hat{v}, v_1\rangle| \geq c$$

where v_1 is the top eigenvector of the matrix $A^{\mathrm{T}}A$.

Moreover, the result holds irrespective of the order in which the vectors $a_1, \ldots, a_n$ are presented to the Oja's algorithm. We will additionally show that even keeping track of the largest norm vector is insufficient to output a vector that has a large correlation with v_1.

Proof. Our instance consists of the following vectors:

1. R copies of the vector $(1/\sqrt{R})e_1$,
2. 1 copy of the vector $(1/\sqrt{R-\varepsilon})e_2$, and
3. α copies of the vector $(1/\sqrt{\alpha \cdot R})e_3$, where $\alpha = 2M$.

Let A be a matrix with rows given by the stream of vectors defined above. We note that the matrix A has rank 3 and the non-zero eigenvalues of the matrix $A^{\mathrm{T}}A$ are $1, 1/(R-\varepsilon), 1/R$ and therefore the *gap* $\lambda_1(A^{\mathrm{T}}A)/\lambda_2(A^{\mathrm{T}}A) = R - \varepsilon$. The top eigenvector of the matrix $A^{\mathrm{T}}A$ is e_1 and the row with the largest norm is $(1/\sqrt{R-\varepsilon})e_2$. Thus, the row with the largest norm is not useful to obtain correlation with the true top eigenvector e_1.

Consider an execution of Oja's algorithm with a learning rate η on the above stream of vectors. The final vector z_n can be written as

$$z_n = \left(I + \frac{\eta}{R}e_1e_1^{\mathrm{T}}\right)^R \left(I + \frac{\eta}{R\alpha}e_3e_3^{\mathrm{T}}\right)^\alpha \left(I + \frac{1}{R-\varepsilon}e_2e_2^{\mathrm{T}}\right) v_0.$$

For $j \in [d]$, let z_{ij} denote the j-th coordinate of the vector z_i so that we have

$$\begin{aligned} z_{n1} &= \left(1 + \frac{\eta}{R}\right)^R \cdot z_{01}, \\ z_{n2} &= \left(1 + \frac{\eta}{R-\varepsilon}\right) \cdot z_{02}, \quad \text{and} \\ z_{n3} &= \left(1 + \frac{\eta}{R\alpha}\right)^\alpha \cdot z_{03}. \end{aligned}$$

We note that $z_{nj} = z_{0j}$ for all $j > 3$. Since $\alpha = 2M$, we have $\eta/R\alpha \le 1/2$ and therefore $(1+\eta/R\alpha) \ge \exp(\eta/2R\alpha)$ and $(1+\eta/R\alpha)^\alpha \ge \exp(\eta/2R)$.

Recall that we want to show that $|\langle z_n, e_1\rangle| < c\|z_n\|_2$ with a large probability. Suppose otherwise and that with probability $\ge 1/10$, we have $|\langle z_n, e_1\rangle| > c\|z_n\|_2 > c\|(0,\ 0,\ 0,\ z_{04},\ \ldots,\ z_{0d})\|_2$.

Since, z_0 is initialized to be a random Gaussian, we have $\|(0,0,0,z_{04},\ldots,z_{0d})\|_2 \ge \sqrt{d}/2$ with probability $1 - \exp(-d)$. Thus, we have with probability $\ge 1/11$ that,

$$|z_{n1}| \ge c\sqrt{d}/2$$

which implies the learning rate must satisfy

$$(1+\eta/R)^R \geq c'\sqrt{d}/2$$

since $|z_{01}| \leq 10$ with probability $\geq 99/100$. Hence, $\eta \geq R((c'd^{1/2})^{1/R} - 1)$. Now consider $|\langle z_n, e_3\rangle|/|\langle z_n, e_1\rangle|$. We have

$$\frac{|\langle z_n, e_3\rangle|}{|\langle z_n, e_1\rangle|} = \frac{\exp(\eta/R)}{(1+\eta/R)^R} \cdot \frac{|z_{03}|}{|z_{01}|}.$$

With probability $\geq 95/100$, we have $1/C \leq |z_{03}|/|z_{01}| \leq C$ for a large enough constant C. We now consider the expression

$$\frac{\exp(\eta/R)}{(1+\eta/R)^R}.$$

The expression is minimized at $\eta = R^2 - R$ and is increasing in the range $\eta \in [R^2 - R, \infty)$. When, $R = O(\log d/\log\log d)$, we have that $R^2 - R \leq R((c'd^{1/2})^{1/R} - 1)$ and therefore for all $\eta \geq R((c'd^{1/2})^{1/R} - 1)$, we have

$$\frac{\exp(\eta/R)}{(1+\eta/R)^R} \geq \frac{\exp((c'd^{1/2})^{1/R})}{e \cdot c'd^{1/2}}.$$

When $R = O(\log d/\log\log d)$, we have

$$\frac{\exp(\eta/R)}{(1+\eta/R)^R} \geq \mathrm{poly}(d)$$

which then implies $|\langle z_n, e_3\rangle| \geq |\langle z_n, e_1\rangle| \cdot \mathrm{poly}(d)/C$ with probability $\geq 95/100$ which contradicts our assumption that $|\langle z_n, e_1\rangle| \geq c\|z_n\|_2$. □

Part III

The Distributed Setting

Chapter 13

Optimal Communication Bounds for Classic Functions in the Coordinator Model

13.1 Introduction

In modern applications data is often distributed across multiple servers and communication is a bottleneck. This motivates minimizing the communication cost for solving classical functions of interest. A standard model of distributed computation is the *coordinator* or *message-passing* model, in which there are s servers, each with an input, and a coordinator with no input. All communication goes through the coordinator, who decides who speaks next. This models arbitrary point-to-point communication up to a multiplicative factor of 2 and an additive $\lceil \log_2(\#\text{ of servers}) \rceil$ bits per message, since the coordinator can forward a message from server i to server j provided i indicates which server should receive the message. The coordinator model is also useful in distributed functional monitoring [CMY11]. Numerous functions have been studied in the coordinator model, such as bitwise operations on vectors [PVZ16], set-disjointness [BEO+13], graph problems [WZ17], statistical problems [WZ17], and many more.

In the coordinator model, we measure the efficiency of a protocol by looking at the following: (i) the overall number of bits of communication required by the protocol and (ii) the number of rounds of communication in the protocol. In each round of communication, each of the servers sends a message to the coordinator based on their input and messages from the coordinator in previous rounds. Based on the messages received from all the servers in this round and earlier rounds, the coordinator sends a possibly distinct message to each of the servers. Thus, in a protocol with one round, each of the servers sends a message to the coordinator based *only* on their inputs and the coordinator has to compute the output based only on these messages. We additionally assume that all the servers and the coordinator have access to a shared source of randomness which they can use to sample shared random variables.

We revisit classical entrywise function approximation, which includes the F_k moment estimation as a special case. Surprisingly, despite the simplicity of the coordinator model and the optimal bounds known for such functions in related models such as the streaming model, the optimal communication complexity of computing or approximating such functions in the coordinator model is open. In the F_k moment estimation problem there are s players, the j-th of which holds a non-negative vector[1] $x(j) \in \mathbb{R}^n$, and the goal is to, with constant probability, output a $(1+\varepsilon)$-multiplicative approximation to $F_k(x) \coloneqq \sum_{j=1}^{n} |x_i|^k$, where $x = \sum_{j=1}^{s} x(j)$. A large body of work has studied F_k-moment estimation in a stream, originating with work of Alon, Matias, and Szegedy [AMS99].

In the coordinator model, Cormode et al. [CMY11] initiated the study of this problem and gave a protocol achieving $\widetilde{O}(n^{1-2/k}\,\mathrm{poly}(s/\varepsilon))$ bits of total communication for $k \geq 2$. They optimize their bound for $k = 2$ and achieve a quadratic dependence on s. They also achieve protocols for $k \in \{0, 1\}$ with a linear dependence on s. We note that their algorithms hold in the more general distributed functional monitoring framework. The upper bound was improved by Woodruff and Zhang to $\widetilde{O}(s^{k-1}/\varepsilon^{\Theta(k)})$ bits in [WZ12], which showed that a polynomial dependence on n is not needed for $k > 2$. Unfortunately the $1/\varepsilon^{\Theta(k)}$ multiplicative factor is prohibitive, and Kannan, Vempala, and Woodruff [KVW14] claimed an improved bound of $\widetilde{O}((s^{k-1}+s^3)/\varepsilon^3)$ bits. However, there appears to be a gap in their analysis which is not clear how to fix [KVW18]. We describe this gap in the appendix. Their algorithm for general function approximation can be used to obtain an algorithm which uses $\widetilde{O}(s^k/\varepsilon^2)$ bits of communication. Thus, the current state of the art is a $\min(\widetilde{O}(s^{k-1}/\varepsilon^{\Theta(k)}), \widetilde{O}(s^k/\varepsilon^2))$ upper bound from [WZ12, KVW14] and the $\Omega(s^{k-1}/\varepsilon^2)$ lower bound of [WZ12]. Even if the work of [KVW14] can be fixed, it would not match the existing lower bounds, and an important open question is:

Question 1: What is the complexity of F_k-estimation in the coordinator model?

As $F_k(x) = \sum_{i=1}^{n} |x_i|^k$ is just one example of an entrywise function $\sum_{i=1}^{n} f(x_i)$ for a non-negative function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, it is natural to ask what the complexity of approximating $\sum_{i=1}^{n} f(x_i)$ is in terms of f. Indeed, a wide body of work originating with that of Braverman and Ostrovsky [BO10] does exactly this for the related data stream model. In the coordinator model, Kannan, Vempala, and Woodruff [KVW14] attempt to characterize the complexity of f by defining a parameter they call $c_{f,s}$, which is the smallest positive number for which $f(y_1 + \cdots + y_s) \leq c_{f,s}(f(y_1) + \cdots + f(y_s))$ for all $y_1, \ldots, y_s \geq 0$.

For general functions f, they give a protocol which uses $O(s^2 c_{f,s}/\varepsilon^2)$ bits of communication up to polylogarithmic factors. Assuming that the function f is super-additive, i.e., $f(y_1 + y_2) \geq f(y_1) + f(y_2)$ for all $y_1, y_2 \geq 0$, their upper bound can be further improved to $O(s c_{f,s}/\varepsilon^2)$. They also give an $\Omega(c_{f,s}/\varepsilon)$ communication lower bound.

We note that a number of interesting entrywise functions have been considered in optimization contexts, such as the M-Estimators (see, e.g., [CW15]), and a natural such estimator is the Huber loss function $f(x) = x^2/(2\tau)$ for $|x| \leq \tau$, and $f(x) = |x| - \tau/2$ otherwise. It is not hard to show

[1]If the vectors are allowed to have negative entries, then there is an $\Omega(n^{1-2/k})$ bit lower bound on the amount of communication when $k > 2$ [BJKS04].

$c_{f,s} = s$ for the Huber loss function, and so the best known upper bound is $\widetilde{O}(s^2/\varepsilon^2)$ bits while the lower bound is only $\Omega(s/\varepsilon)$. Given the gap between the upper and lower bounds, it is unclear if the parameter $c_{f,s}$ captures the complexity of approximating the sum $\sum_i f(x_i)$. We observe that the communication complexity of the problem is better captured by a new parameter $c_f[s]$ defined as the smallest number for which

$$f(y_1 + \cdots + y_s) \le \frac{c_f[s]}{s}(\sqrt{f(y_1)} + \cdots + \sqrt{f(y_s)})^2 \quad \text{for all} \quad y_1, \ldots, y_s \ge 0. \tag{13.1}$$

Since, $(\sqrt{f(y_1)} + \cdots + \sqrt{f(y_s)})^2 \ge f(y_1) + \cdots + f(y_s)$, we obtain $c_f[s] \le c_{f,s} \cdot s$ and using the Cauchy-Schwarz inequality, we can show that $c_{f,s} \le c_f[s]$. We consider the following question:

Question 2: What is the complexity of entrywise function approximation in the coordinator model? Can one characterize the complexity completely in terms of $c_f[s]$?

13.1.1 Our Results

To answer Question 2, we give a two round protocol for approximating $\sum_i f(x_i)$ for non-negative functions which have an "approximate-invertibility" property.

Definition 13.1.1 (Approximate Invertibility)**.** We say that a function f satisfies approximate invertibility with parameters $\theta, \theta', \theta'' > 1$ if all the following properties hold: (i) *super-additivity*: $f(x+y) \ge f(x) + f(y)$ for all $x, y \ge 0$, (ii) for all $y \ge 0$, $f(\theta' y) \ge \theta \cdot f(y)$, and (iii) for all $y \ge 0$, $f(y/4 \cdot \sqrt{\theta} \cdot \theta') \ge f(y)/\theta''$.

The super-additivity and the fact that $f(y) \ge 0$ for all y implies that $f(0) = 0$. We note that any increasing convex function f with $f(0) = 0$ satisfies the super-additivity property and hence it is not a very strong requirement. It is satisfied by $f(x) = x^k$ for $k \ge 1$ and the Huber loss function with any parameter. For such functions, we prove the following theorem:

Theorem 13.1.2 (Informal, Theorem 13.4.15)**.** *Let there be s servers with the j-th server holding a non-negative n-dimensional vector $x(j)$, and define $x = x(1)+\cdots+x(s)$. Given a function f which satisfies the approximate invertibility property with parameters $\theta, \theta', \theta'' > 1$, our two round protocol approximates $\sum_i f(x_i)$ up to a $1 \pm \varepsilon$ factor with probability $\ge 9/10$. Our protocol uses a total communication of $O_{\theta,\theta',\theta''}(c_f[s]/\varepsilon^2)$ bits up to polylogarithmic factors in the dimension n.*

For $f(x) = x^k$, $k \ge 2$, we see that $c_f[s] = s^{k-1}$ and can take $\theta = 2$, $\theta' = 2^{1/k}$ and $\theta'' = 2 \cdot 8^{k/2}$. Hence, our algorithm uses s^{k-1}/ε^2 bits of total communication up to multiplicative factors depending on k and $\log n$, thus matching the known lower bounds from [WZ12]. We additionally show that any one-round algorithm must use $\Omega(s^{k-1}/\varepsilon^k)$ bits of communication and hence our protocol achieves the optimal communication bounds using the fewest possible number of rounds, thus resolving Question 1 completely. We summarize the results for F_k-moment estimation in Table 13.1.

We can also use our protocol to approximate *higher-order correlations* studied by Kannan, Vempala

F_k estimation algorithm	Upper Bound	Lower Bound
2-round algorithm	$\widetilde{O}(s^{k-1}/\varepsilon^2)$ (Corollary 13.4.16)	$\Omega(s^{k-1}/\varepsilon^2)$ [WZ12]
1-round algorithm	$\widetilde{O}(s^{k-1}/\varepsilon^{\Theta(k)})$ [WZ12]	$\widetilde{\Omega}(s^{k-1}/\varepsilon^k)$ (Theorem 13.5.3)

Table 13.1: Upper and lower bounds on the total communication for F_k approximation in the coordinator model. As mentioned, the claimed $\widetilde{O}(\varepsilon^{-3}(s^{k-1}+s^3))$ upper bound for F_k in [KVW14] has a gap.

and Woodruff [KVW14]. In this problem, each server j holds a set of non-negative vectors W_j and given functions $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ and $g : \mathbb{R}^k_{\geq 0} \to \mathbb{R}_{\geq 0}$, the correlation $M(f,g)$ is defined as

$$M(f,g,W_1,\ldots,W_k) \coloneqq \sum_{i_1,\ldots,i_k \text{ distinct}} f\left(\sum_j \sum_{v \in W_j} g(v_{i_1},\ldots,v_{i_k})\right). \tag{13.2}$$

Kannan, Vempala and Woodruff [KVW14] note that this problem has numerous applications and give some examples. Our protocol for estimating $\sum_i f(x_i)$ in the coordinator model extends in a straightforward way to the problem of estimating higher-order correlations. We show the following result:

Theorem 13.1.3 (Informal, Theorem 13.4.17)**.** *Let there be s servers with the j-th server holding a set of n-dimensional non-negative vectors W_j. Given a function f that has the approximate invertibility property with parameters $\theta, \theta', \theta'' > 1$ and a function $g : \mathbb{R}^k_{\geq 0} \to \mathbb{R}_{\geq 0}$, our randomized two round protocol approximates $M(f,g,W_1,\ldots,W_k)$ up to a $1 \pm \varepsilon$ factor with probability $\geq 9/10$. The protocol uses a total of $O_{\theta,\theta',\theta''}\left(c_f[s]\,\mathrm{poly}(k,\log n)/\varepsilon^2\right)$ bits of communication.*

Our algorithm for approximating $\sum_i f(x_i)$ in the coordinator model is inspired by a one round protocol for sampling from an "additively-defined distribution", which can also approximate the sampling probability of the index that was sampled. This protocol lets us sample, in one round, from very general distributions such as the leverage scores. Our result is stated in the following theorem.

Theorem 13.1.4 (Informal, Theorem 13.3.2)**.** *Given that each server j has a non-negative vector $p(j) \in \mathbb{R}^n$, define $q_i \coloneqq \sum_j p_i(j)$. There is a randomized algorithm which outputs FAIL with probability $\leq \frac{1}{\mathrm{poly}(n)}$ and conditioned on not outputting FAIL, it outputs a coordinate $\hat{i}$ along with a value $\hat{q}$ such that for all $i \in [n]$*

$$\mathbf{Pr}[\hat{i} = i \text{ and } \hat{q} \in (1 \pm O(\varepsilon))\frac{q_i}{\sum_i q_i}] = (1 \pm O(\varepsilon))\frac{q_i}{\sum_i q_i} \pm \frac{1}{\mathrm{poly}(n)}.$$

The algorithm uses one round and has a total communication of $O(s\,\mathrm{polylog}(n)/\varepsilon^2)$ words.

13.1.2 Our Techniques

Sampling from *Additively-Defined* Distributions. At the heart of our results for approximating $\sum_{i=1}^{n} f(x_i) = \sum_{i=1}^{n} f(\sum_{j=1}^{s} x_i(j))$ is a general technique to sample from an "additively-defined" distribution using only one round of communication. To obtain our tightest communication bounds, our algorithm for approximating $\sum_{i=1}^{n} f(x_i)$ does not use this protocol in a black-box way, but the techniques used are similar to the ones used in this protocol. In this setting, the j-th server holds a non-negative vector $p(j) \in \mathbb{R}_{\geq 0}^{n}$ and the coordinator wants to sample from a distribution over $[n]$ with the probability of sampling $i \in [n]$ being proportional to $q_i = p_i(1) + \cdots + p_i(s)$.

Additionally, if the coordinate i is sampled by the coordinator, the coordinator also needs to be able to estimate the probability with which i is sampled. This is an important requirement to obtain (approximately) unbiased estimators in applications involving importance sampling. If the coordinator just wants to sample from the distribution, it can run the following simple protocol: first, each server j samples a coordinate $\boldsymbol{i}_j$ from its local distribution, i.e., $\mathbf{Pr}[\boldsymbol{i}_j = i] = p_i(j)/\sum_i p_i(j)$. Each server sends the coordinate $\boldsymbol{i}_j$ along with the quantity $\sum_i p_i(j)$ to the coordinator. The coordinator then samples a random server $\boldsymbol{j}$ from the distribution $\mathbf{Pr}[\boldsymbol{j} = j] = (\sum_i p_i(j))/\sum_{j'}(\sum_i p_i(j'))$ and then takes $\boldsymbol{i}_{\boldsymbol{j}}$, i.e., the coordinate sent by the server $\boldsymbol{j}$, to be the sample. We note that

$$\mathbf{Pr}[\boldsymbol{i}_{\boldsymbol{j}} = i] = \sum_{j} \mathbf{Pr}[\boldsymbol{j} = j]\, \mathbf{Pr}[\boldsymbol{i}_j = i] = \sum_{j} \frac{\sum_{i'} p_{i'}(j)}{\sum_j \sum_{i'} p_{i'}(j)} \cdot \frac{p_i(j)}{\sum_{i'} p_{i'}(j)} = \frac{\sum_j p_i(j)}{\sum_j \sum_{i'} p_{i'}(j)} = \frac{q_i}{\sum_{i'} q_{i'}}.$$

Hence, the distribution of $\boldsymbol{i}_{\boldsymbol{j}}$ is *correct*. But notice that there is no easy way for the coordinator to estimate the probability of sampling $\boldsymbol{i}_{\boldsymbol{j}}$ since it may not receive any information about this coordinate from the other servers. Therefore, to obtain the probability with which $\boldsymbol{i}_{\boldsymbol{j}}$ was sampled, the coordinator needs another round of communication, which we wish to avoid.

We will now give a protocol that can also approximate the sampling probabilities with only one round of communication. The protocol we describe here is a simpler version of the full protocol in Section 13.3. Consider the following way of sampling from the distribution in which i has a probability proportional to q_i. Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ be independent standard exponential random variables. Let $i^* = \arg\max_{i \in [n]} \boldsymbol{e}_i^{-1} q_i = \arg\max_{i \in [n]} \boldsymbol{e}_i^{-1}(p_i(1) + \cdots + p_i(s))$. By standard properties of the exponential random variable, we have $\mathbf{Pr}[i^* = i] = \frac{\sum_j p_i(j)}{\sum_j \sum_{i'} p_{i'}(j)} = \frac{q_i}{\sum_{i'} q_{i'}}$.

Hence, the random variable i^* also has the *right* distribution. The advantage now is that we can additionally show that with a high probability, $\sum_i \boldsymbol{e}_i^{-1} \left(\sum_j p_i(j)\right) \leq (C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} \sum_j p_{i^*}(j)$. In other words, if we define s vectors $r(1), \ldots, r(s)$, one at each of the servers, such that $r_i(j) = \boldsymbol{e}_i^{-1} p_i(j)$ and define $r = \sum_{j=1}^{s} r(j)$, then the coordinate r_{i^*} is a $1/(C \log^2 n)$ ℓ_1 *heavy hitter*, as in, $r_{i^*} \geq \|r\|_1/(C \log^2 n)$.

Suppose a deterministic sketch matrix $S \in \mathbb{R}^{m \times n}$ is an α-incoherent matrix for $\alpha = \varepsilon/(4C \log^2 n)$. Here we say that a matrix S is α-incoherent if all the columns of S have unit Euclidean norm and for any $i \neq i' \in [n]$, $|\langle S_{*i}, S_{*i'} \rangle| \leq \alpha$. Nelson, Nguyen and Woodruff [NNW14] give constructions of such matrices with $m = O(\log(n)/\alpha^2)$ rows and show that for any vector x, $\|x - S^{\mathrm{T}}Sx\|_\infty \le \alpha\|x\|_1$.

Now suppose that each server j computes the vector $S \cdot r(j)$ and uses $O(\mathrm{polylog}(n)/\varepsilon^2)$ words of communication to send the vector $S \cdot r(j)$ to the coordinator. The coordinator receives the vectors $S \cdot r(1), \ldots, S \cdot r(s)$ and computes the vector $S \cdot r = S \cdot r(1) + \cdots + S \cdot r(s)$ and can then compute a vector $r' = S^{\mathrm{T}}Sr$ satisfying $\|r - r'\|_\infty \le \alpha\|r\|_1$.

Conditioned on the event that $\|r\|_1 \le (C \log^2 n) \cdot r_{i^*}$, as α is set to be $\varepsilon/(4C\log^2 n)$, we obtain that $r'_{i^*} = (1 \pm \varepsilon/4) \cdot r_{i^*}$ and for all $i \neq i^*$, we have $r'_i \le r_i + (\varepsilon/4) \cdot r_{i^*}$. Using properties of exponential random variables, we can also show that with probability $\ge 1 - O(\varepsilon)$, $\max_i r_i \ge (1+\varepsilon) \cdot$ second-$\max_i r_i$. Conditioned on this event as well, we obtain that for all $i \neq i^*$, $r'_i < r_{i^*}(1 - \varepsilon/4) \le r'_{i^*}$. Hence, the largest coordinate in the vector r' is exactly i^* and the value r_{i^*} can be recovered up to a $1 \pm \varepsilon/4$ factor. Overall, the coordinator can find a coordinate $\hat{i}$ and a value $\hat{q} = \boldsymbol{e}_{\hat{i}} \cdot (r_{\hat{i}})'$ such that for all $i \in [n]$,

$$\mathbf{Pr}[\hat{i} = i \text{ and } \hat{q} \in (1 \pm \varepsilon/4)q_i] = \frac{q_i}{\sum_{i'} q_{i'}} \pm O(\varepsilon). \tag{13.3}$$

Thus this procedure lets us sample from a distribution close to that of the desired distribution while at the same time lets us approximate the probability of the drawn sample. In Section 13.3, we give a protocol, which instead of using incoherent matrices, uses an ℓ_1 sampling based algorithm to compute the coordinate i^* and approximate the value q_{i^*}. We end up obtaining a sample $\hat{i} \in [n]$ and a value $\hat{q}$ for which

$$\mathbf{Pr}[\hat{i} = i \text{ and } \hat{q} = (1 \pm \varepsilon)\frac{q_i}{\sum_{i'} q_{i'}}] = (1 \pm \varepsilon)\frac{q_i}{\sum_{i'} q_{i'}} \pm \frac{1}{\mathrm{poly}(n)}. \tag{13.4}$$

Additionally, computing the coordinate $\hat{i}$ does not require $\Omega(n)$ time at the coordinator using the protocol in Section 13.3.

Function Sum Approximation. Our aim is to obtain an algorithm which approximates the sum $\sum_{i=1}^n f(x_i) = \sum_{i=1}^n f(\sum_{j=1}^s x_i(j))$ up to a $1 \pm \varepsilon$ factor for a non-negative, super-additive function f. The protocol described above for sampling from additively-defined distributions shows that correlating randomness across all the servers using exponential random variables is a powerful primitive in this context.

In the function sum approximation problem, each server holds a nonnegative vector $x(1), \ldots, x(s)$, respectively, and the coordinator wants to approximate $\sum_i f(x_i) = \sum_i f(\sum_j x_i(j))$. Suppose that each server j defines a vector $p(j) \in \mathbb{R}^n$ such that $p_i(j) = f(x_i(j))$. Then the above described protocol can be used to sample approximately from the distribution in which i has probability $(1 \pm \varepsilon)\frac{f(x_i(1)) + \cdots + f(x_i(s))}{\sum_{i'}\sum_j f(x_{i'}(j))} \pm \frac{1}{\mathrm{poly}(n)}$. Using super-additivity and the definition of the parameter $c_f[s]$, we obtain that the above probability is at least $(1 \pm \varepsilon)\frac{f(x_i)}{c_f[s] \cdot \sum_{i'} f(x_{i'})} \pm \frac{1}{\mathrm{poly}(n)}$. This distribution is off by a multiplicative $c_f[s]$ factor from the distribution we need to sample coordinates from in order to estimate $\sum_i f(x_i)$ with a low variance. Thus, we need $O(c_f[s]/\varepsilon^2)$ samples from the above distribution, and this overall requires a communication of $O(s \cdot c_f[s]/\varepsilon^4)$ bits, which is more than the total communication required by the protocol of [KVW14].

Additionally, when the coordinate i is sampled, the protocol lets us estimate $f(x_i(1)) + \cdots + f(x_i(s))$, but the quantity we want to construct an estimator for is the value $f(x_i(1) + \cdots + x_i(s))$, which requires an additional round of communication and defeats the point of obtaining a protocol that can approximate the sampling probability in the same round. Overall, a protocol based on this procedure requires $O(s \cdot c_f[s] \cdot \mathrm{polylog}(n)/\varepsilon^4)$ bits of communication and two rounds of communication.

Thus, we need a different technique to obtain algorithms which can approximate $\sum_i f(x_i)$ more efficiently. We continue to use exponential random variables to correlate the randomness across the servers but instead heavily use the max-stability property to design our protocol. Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ be independent standard exponential random variables. The max-stability property asserts that for any $f_1, \ldots, f_n \geq 0$, the random variable $\max(f_1/\boldsymbol{e}_1, \ldots, f_n/\boldsymbol{e}_n)$ has the same distribution as $(\sum_{i=1}^n f_i)/\boldsymbol{e}$ where $\boldsymbol{e}$ is also a standard exponential random variable. Now using the *median* of $O(1/\varepsilon^2)$ independent copies of the random variable $(\sum_{i=1}^n f_i)/\boldsymbol{e}$, we can compute an approximation of $\sum_{i=1}^n f_i$ up to a $1 \pm \varepsilon$ factor with high probability. Thus, in the coordinator model, if there is a protocol that can find the value of the random variable $\max_i f(x_i)/\boldsymbol{e}_i$, then we can use it to compute a $1 \pm \varepsilon$ approximation to $\sum_i f(x_i)$ by running the protocol for $O(1/\varepsilon^2)$ independent copies of the exponential random variables. From here on, we explain how we construct such a protocol.

Given the exponential random variables $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$, define $i^* \coloneqq \arg\max_{i \in [n]} \boldsymbol{e}_i^{-1} f(x_i)$. As mentioned above we would like to find the value of $\boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$. The "heavy-hitter" property we used previously shows that with probability $\geq 1 - \frac{1}{\mathrm{poly}(n)}$,

$$\sum_i \boldsymbol{e}_i^{-1} f(x_i) \leq (C \log^2 n) \cdot \max_i \boldsymbol{e}_i^{-1} f(x_i). \tag{13.5}$$

This is the main property that leads to a communication efficient algorithm that can identify the max coordinate i^* and the value x_{i^*}. From here on, condition on the above event.

Note that we are shooting for a protocol that uses at most $O(c_f[s] \cdot \mathrm{polylog}(n))$ bits of total communication and succeeds in computing the value $\boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$ with a probability $\geq 1 - \frac{1}{\mathrm{poly}(n)}$. Fix a server $j \in [s]$. Consider the random variable $\boldsymbol{i}$ which takes values in the set $[n]$ according to the distribution $\mathbf{Pr}[\boldsymbol{i} = i] = \boldsymbol{e}_i^{-1} f(x_i(j)) / \sum_{i \in [n]} \boldsymbol{e}_i^{-1} f(x_i(j))$. Since the server j knows all the values, $x_1(j), x_2(j), \ldots, x_n(j)$, it can compute the above probability distribution and can sample N (for a value to be chosen later) independent copies $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_N$ of the random variable $\boldsymbol{i}$. Let $\mathsf{SC}_j \coloneqq \{\, \boldsymbol{i}_1, \ldots, \boldsymbol{i}_N \,\}$[2] be the set of coordinates sampled by server j. Server j then sends the set SC_j along with the values $x_i(j)$ for $i \in \mathsf{SC}_j$. Note that the coordinator can compute $f(x_i(j))$ since it knows the

[2]We use the notation SC_j since it denotes the "**S**ampled **C**oordinates" at server j.

definition of the function f. The total communication from all the servers to the coordinator until this point is $O(s \cdot N)$ words.

Now define $\mathsf{SC} \coloneqq \bigcup_j \mathsf{SC}_j$ to be the set of coordinates that is received by the central coordinator from all the servers. We will first argue that $i^* \in \mathsf{SC}$ with a large probability if the number of sampled coordinates at each server $N = \Omega(c_f[s] \cdot \text{polylog}(n)/s)$. To prove this, we use the fact that $\boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$ is significantly large since we conditioned on the event in (13.5), and then apply the definition (13.1) of the parameter $c_f[s]$.

Conditioned on the event that the coordinate $i^* \in \mathsf{SC}$, a simple algorithm to determine the value of $\boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$ would be for the coordinator to query the value of $x_i(j)$ for all $i \in \mathsf{SC}$ from all the servers j. Unfortunately, this requires a total communication of $\Omega(s \cdot c_f[s] \cdot \text{polylog}(n))$ bits since the set SC could have a size as large as $s \cdot N = \Omega(c_f[s] \cdot \text{polylog}(n))$. As we are aiming for a protocol that uses about $O(c_f[s] \cdot \text{polylog}(n))$ bits of total communication, the coordinator cannot ask for the values of all the coordinates in the set SC.

If the coordinator finds a smaller subset PL[3] $\subseteq \mathsf{SC} \subseteq [n]$ that contains i^*, with a size $|\mathsf{PL}|$ of about polylog(n), the coordinator can then query for $x_i(j)$ for $i \in \mathsf{PL}$ for all j using only a communication of $O(s \cdot \text{polylog}(n)) = O(c_f[s] \cdot \text{polylog}(n))$ bits since $c_f[s] \geq s$.

From here on condition on the event that $i^* \in \mathsf{SC}$. To find such a small subset PL, our strategy is to construct $\hat{x}_i$ for each $i \in \mathsf{SC}$ so that the following properties are simultaneously satisfied with probability $\geq 1 - \frac{1}{\text{poly}(n)}$: (i) for all $i \in \mathsf{SC}$, $\hat{x}_i \leq x_i$ and (ii) $\boldsymbol{e}_{i^*}^{-1} f(\hat{x}_{i^*}) \geq \alpha \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$ for some value $\alpha < 1$. Define $\mathbf{Est}_i \coloneqq \boldsymbol{e}_i^{-1} f(\hat{x}_i)$ for $i \in \mathsf{SC}$. If the constructed values $\hat{x}_i$ for $i \in \mathsf{SC}$ satisfy these two properties, we have that $\mathbf{Est}_i \leq \boldsymbol{e}_i^{-1} f(x_i)$ for all i and $\mathbf{Est}_{i^*} \geq \alpha \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$ by monotonicity of f.

Recall that we conditioned on the event $\sum_i \boldsymbol{e}_i^{-1} f(x_i) \leq (C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$ which then implies that the number of coordinates i, with $\mathbf{Est}_i \geq \mathbf{Est}_{i^*}$ is at most $(C \log^2 n)/\alpha$. Now, if we define PL to be the set of coordinates $i \in \mathsf{SC}$ with the $C \log^2 n/\alpha$ largest values, then we have that $i^* \in \mathsf{PL}$. The coordinator can then determine $\boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$ after a second round of communication from the servers in which it asks for the values of $x_i(j)$ from all the servers j only for the coordinates $i \in \mathsf{PL}$.

Fix a coordinate $i \in \mathsf{SC}$. We will briefly describe how $\hat{x}_i$ is computed by the coordinator using only the information it receives in the first round of communication from the servers. We say $x_i(j)$ is the *contribution* of server j to x_i. If $i \neq i^*$, then we can safely ignore the contribution from any number of servers to x_i when we are trying to construct the estimator $\hat{x}_i$. However, the coordinator does not know what i^* is and cannot arbitrarily drop the contribution from servers when trying to compute the estimator $\hat{x}_i$.

We first define two disjoint subsets of servers $\textsc{Large}_i$ and $\textsc{Small}_i$: we put $j \in \textsc{Large}_i$ if the probability of the coordinate i being sampled at j is *very high*, and we put $j \in \textsc{Small}_i$ if the probability is *very low*. Since the probability of i being sampled at the servers in $\textsc{Large}_i$ is very large, we can union bound over all $i \in [n]$ and all servers $j \in \textsc{Large}_i$ and assume that it does happen, and can therefore estimate the contribution of all the servers in $\textsc{Large}_i$ exactly.

[3]We use PL to denote that these set of coordinates have "**Probably Large**" values of $\boldsymbol{e}_i^{-1} f(x_i)$.

We then argue that the contribution from all the servers $j \in \textsc{Small}_i$ can be "ignored": by ignoring the contribution of a set of servers S_i, we mean that $\boldsymbol{e}_{i^*}^{-1} f(x_{i^*} - \sum_{j \in S_{i^*}} x_{i^*}(j))$ is a significant portion of $\boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$. Note that we only need to care about how excluding the contribution of S_{i^*} to x_{i^*} affects our ability in obtaining $\hat{x}_{i^*}$ which satisfies the above property, and we need not care about the effects of excluding the contribution to x_i from the set of servers S_i for all other $i \neq i^*$ since we are only trying to underestimate such x_i.

We then need to estimate the contribution to x_i from the "intermediate" servers, i.e., those that are neither $\textsc{Large}$ nor $\textsc{Small}$. We bucket the servers j based on the values $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$ and $\boldsymbol{e}_i^{-1} f(x_i(j)) / \sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$ (the probability that a given sample at server j is equal to i). We show that the *size* of a bucket is enough to approximate the contribution from all the servers in the bucket towards x_i and argue that if the size is not very large, then the contribution from the bucket can be ignored in the sense described above. Of course, the coordinator can not determine the size of a bucket but given that a server j sampled the coordinate i, the coordinator can compute which bucket the server j belongs to. We show that when the size of the bucket is large enough, the number of servers in the bucket that sample the coordinate i is concentrated enough that we can estimate its size. We thus identify which buckets have a large size based on the number of servers in the bucket that sample i, and for each bucket that we identify as large, we approximate the size up to constant factor. This can then be used to approximate the contribution of the bucket to x_i, and hence obtain the estimator $\hat{x}_i$.

This wraps up our protocol for computing $\max_i \boldsymbol{e}_i^{-1} f(x_i)$, with a high probability, and using a total of $O(c_f[s] \cdot \mathrm{polylog}(n))$ bits of total communication across two rounds. Running this protocol concurrently for $O(1/\varepsilon^2)$ copies of the exponential random variables, we can then obtain a $1 \pm \varepsilon$ approximation to $\sum_i f(x_i)$ with high probability.

13.2 Preliminaries

13.2.1 Notation

We use the notation $F_k(x)$ for $\|x\|_k^k$. For a parameter k, we use the notation $O_k(f)$ to hide the multiplicative factors that depend purely on k.

13.2.2 Exponential Random Variables

We use the following properties of exponential random variables extensively throughout this chapter.

1. If $\boldsymbol{e}$ is a standard exponential random variable, then $\mathbf{Pr}[\boldsymbol{e} \geq C \log n] = 1/n^C$ and $\mathbf{Pr}[\boldsymbol{e} \leq t] \leq t$ for any $C, t \geq 0$.

2. If $f_1, \ldots, f_n \geq 0$ are arbitrary and $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ are standard exponential random variables, then $\max_i \boldsymbol{e}_i^{-1} f_i$ has the same distribution as $\boldsymbol{e}^{-1}(\sum_i f_i)$. This property is referred to as "max-stability". This shows that with probability $\geq 1 - 1/n^C$, $\max_i \boldsymbol{e}_i^{-1} f_i \geq (\sum_i f_i)/C \log n$.
3. In the same setting, if $i^* = \arg\max_i \boldsymbol{e}_i^{-1} f_i$, then $\mathbf{Pr}[i^* = i] = f_i / \sum_j f_j$.
4. Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t$ are t independent standard exponential random variables. If $t = \Omega(1/\varepsilon^2)$, then

$$\mathbf{Pr}[(1-\varepsilon)\ln(2) \leq \text{median}(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t) \leq (1+\varepsilon)\ln(2)] \geq 9/10.$$

   Hence, for any $F > 0$, $\text{median}(F/\boldsymbol{e}_1, \ldots, F/\boldsymbol{e}_t) \cdot \ln(2) \in [(1-\varepsilon)F, (1+\varepsilon)F]$ with probability $\geq 9/10$.

We use the following lemma extensively which shows that with a probability $\geq 1 - 1/\text{poly}(n)$, $\sum_i \boldsymbol{e}_i^{-1} f_i \leq (C \log^2 n) \max_i \boldsymbol{e}_i^{-1} f_i$.

Lemma 13.2.1. *Given any $f_1, \ldots, f_n \geq 0$, let $F = \sum_{i=1}^n f_i$. With probability $\geq 1 - 1/\text{poly}(n)$,*

$$\frac{\max f_i/\boldsymbol{e}_i}{\sum_{i=1}^n f_i/\boldsymbol{e}_i} \geq \frac{1}{C \log^2 n}.$$

Proof. Condition on the event that $1/n^3 \leq \boldsymbol{e}_i \leq 3 \log n$ for all i. The event has a probability of at least $1 - 2/n^2$ by a union bound. Let the event be denoted $\mathcal{E}$. Conditioned on $\mathcal{E}$, we have $\max_i f_i/\boldsymbol{e}_i \leq n^3 F$. Now consider the interval $I = [F/(Cn \log n), n^3 F]$. The values $f_i/\boldsymbol{e}_i$ that are smaller than $F/(Cn \log n)$ contribute at most $F/(C \log n)$ to the sum $\sum_{i=1}^n f_i/\boldsymbol{e}_i$.

Partition the interval I into $I_1, \ldots,$ such that $I_j = [2^{j-1}(F/(Cn \log n)), 2^j(F/(Cn \log n)))$. Note that there are at most $O(\log n)$ such intervals. We will use the fact that conditioned on $\mathcal{E}$, the random variables $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ are still independent.

Let $\boldsymbol{X}_j$ be the number of indices i such that $f_i/\boldsymbol{e}_i \in I_j$. We have

$$\sum_i f_i/\boldsymbol{e}_i \leq \frac{F}{C \log n} + \sum_{j=0}^{O(\log n)} 2^j \boldsymbol{X}_j \frac{F}{Cn \log n}.$$

Let $\boldsymbol{Y}_{ij} = 1$ if $f_i/\boldsymbol{e}_i \geq 2^{j-1}(F/(Cn \log n))$ and $\boldsymbol{Y}_{ij} = 0$ otherwise. Note that $\boldsymbol{X}_j \leq \sum_{i=1}^n \boldsymbol{Y}_{ij}$ and that for a fixed j, the random variables $\boldsymbol{Y}_{ij}$ are mutually independent given $\mathcal{E}$. We have

$$\mathbf{Pr}[\boldsymbol{Y}_{ij} = 1 \mid \mathcal{E}] = \mathbf{Pr}\left[\boldsymbol{e}_i \leq \frac{Cn \log n}{2^{j-1}} \frac{f_i}{F} \mid \mathcal{E}\right] \leq \frac{2Cn \log n}{2^{j-1}} \frac{f_i}{F}$$

since the p.d.f. of the exponential distribution is bounded above by 1 and $\mathbf{Pr}[\mathcal{E}] \geq 1/2$. By linearity of expectation, $\mathbf{E}[\sum_i \boldsymbol{Y}_{ij} \mid \mathscr{E}] \le 2Cn\log n/(2^{j-1})$. By Bernstein's inequality, we get

$$\mathbf{Pr}[\sum_i \boldsymbol{Y}_{ij} \ge (\mathbf{E}[\sum_i \boldsymbol{Y}_{ij} \mid \mathscr{E}] + t) \mid \mathscr{E}] \le \exp\left(-\frac{t^2/2}{2Cn\log n/(2^{j-1}) + t/3}\right).$$

Setting $t = 2Cn\log n/(2^{j-1}) + 6\log n$, we obtain that

$$\mathbf{Pr}[\sum_i \boldsymbol{Y}_{ij} \ge (\mathbf{E}[\sum_i \boldsymbol{Y}_{ij} \mid \mathscr{E}] + t) \mid \mathscr{E}] \le \frac{1}{n^2}.$$

We union bound over all j and obtain that for all j,

$$\boldsymbol{X}_j \le \frac{2Cn\log n}{2^{j-1}} + 6\log n.$$

Additionally, we note that if $\boldsymbol{X}_j \ne 0$, then $2^j(F/(Cn\log n)) \le 2\max_i f_i/\boldsymbol{e}_i$. Now,

$$\begin{aligned}\sum_i f_i/\boldsymbol{e}_i \le \frac{F}{C\log n} + \sum_{j:\boldsymbol{X}_j\ne 0} 2^j \boldsymbol{X}_j \frac{F}{Cn\log n} &\le \sum_{j:\boldsymbol{X}_j\ne 0}\left(\frac{2Cn\log n}{2^{j-1}}\frac{2^jF}{Cn\log n} + (6\log n)\max_i f_i/\boldsymbol{e}_i\right)\\ &\le O(F\log n) + O(\log^2 n)\max_i f_i/\boldsymbol{e}_i.\end{aligned}$$

As $\max_i f_i/\boldsymbol{e}_i \ge F/(3\log n)$ with probability $\ge 1 - 1/n^2$ conditioned on $\mathscr{E}$, we get that with probability $\ge 1 - 1/\text{poly}(n)$,

$$\frac{\max_i f_i/\boldsymbol{e}_i}{\sum_i f_i/\boldsymbol{e}_i} \ge \frac{1}{C\log^2 n}.$$ □

If $\max_i f_i/\boldsymbol{e}_i \ge (\sum_i f_i/\boldsymbol{e}_i)/C\log^2 n$, we obtain that for any parameter $T \ge 1$,

$$|\{i \mid f_i/\boldsymbol{e}_i \ge (1/T)\max_i f_i/\boldsymbol{e}_i\}| \le TC\log^2 n$$

which shows that there are only a small number of indices for which $f_i/\boldsymbol{e}_i$ is comparable to $\max_i f_i/\boldsymbol{e}_i$.

13.3 Sampling from Additively-Defined Distributions

Consider the setting of s servers with a coordinator. Assume that the j-th server has a non-negative vector $p(j) \in \mathbb{R}^n$ and the coordinator wants N independent samples from the distribution supported on $[n]$ with the probability of $i \in [n]$ being

$$q_i \coloneqq \frac{\sum_j p_i(j)}{\sum_{i,j} p_i(j)}. \tag{13.6}$$

For each sample, we would ideally also want an estimate to the probability of obtaining that sample. The sampling probabilities are necessary to scale the statistics appropriately to obtain unbiased estimators. A simple one round algorithm for sampling from such a distribution is for all the servers j to send the value $\sum_i p_i(j)$ and an index i with probability proportional to $p_i(j)$. The coordinator then picks a server j with probability proportional to $\sum_i p_i(j)$ and chooses the index i that the server j sampled. Note that the coordinate i sampled this way has the desired distribution. While this sampling procedure only requires one round of communication, note that the coordinator does not have the value $\sum_j p_i(j)$ nor even a way to estimate it. Hence, if the index i was sampled by the coordinator, it is not clear how to compute (or even approximate) q_i without further rounds of communication. If another round of communication is allowed, the coordinator can send the sampled coordinate i to all the servers which in turn report the values $p_i(j)$ from which the coordinator can compute q_i exactly.

Using exponential random variables, we show that there is a protocol using which the coordinator can sample a coordinate i and approximate the probability q_i with only one round of communication.

Let $\boldsymbol{e}_1, \dots, \boldsymbol{e}_n$ be independent exponential random variables that all the servers (and the coordinator) sample using the shared randomness. Each server j locally computes $\boldsymbol{e}_i^{-1} p_i(j)$. Let $i^* \coloneqq \arg\max_i \boldsymbol{e}_i^{-1} \sum_j p_i(j)$. As we have seen, the probability distribution of i^* is exactly as in (13.6) and the advantage now is that with probability $\geq 1 - 1/\text{poly}(n)$,

$$\frac{\boldsymbol{e}_{i^*}^{-1} \sum_j p_{i^*}(j)}{\sum_{i,j} \boldsymbol{e}_i^{-1} p_i(j)} \geq \frac{1}{C \log^2 n}.$$

We can further show the following lemma which is helpful to isolate the max coordinate i^*.

Lemma 13.3.1. *Let $f_1, \dots, f_n \geq 0$ and $i^* = \arg\max_{i \in [n]} \boldsymbol{e}_i^{-1} f_i$ where $\boldsymbol{e}_1, \dots, \boldsymbol{e}_n$ are independent standard exponential random variables. We have for all $i \in [n]$ that*

$$\frac{f_i}{\sum_{i'} f_{i'}} \geq \mathbf{Pr}\left[i^* = i \textit{ and } \boldsymbol{e}_{i^*}^{-1} f_{i^*} \geq (1+\varepsilon) \max_{i' \neq i^*} \boldsymbol{e}_i^{-1} f_{i'} \right] \geq \frac{f_i}{(1+\varepsilon) \sum_{i'} f_{i'}}.$$

Proof. Fix $i \in [n]$. The probability in the theorem statement is equivalent to the probability of $\boldsymbol{e}_i^{-1} f_i \geq (1+\varepsilon) \max_{i' \neq i} \boldsymbol{e}_{i'}^{-1} f_{i'}$. The probability is clearly at most $f_i / \sum_{i'} f_{i'}$.

By min-stability of exponential random variables, $\max_{i' \neq i} \boldsymbol{e}_{i'}^{-1} f_{i'}$ is distributed as $\boldsymbol{e}^{-1} \sum_{i' \neq i} f_{i'}$.

Note that the exponential random variable $\boldsymbol{e}$ is independent of $\boldsymbol{e}_i$. By standard arguments,

$$\mathbf{Pr}[\boldsymbol{e}_i^{-1} f_i \geq \boldsymbol{e}^{-1}(1+\varepsilon)\sum_{i' \neq i} f_i'] = \frac{f_i}{f_i + (1+\varepsilon)\sum_{i' \neq i} f_{i'}} \geq \frac{f_i}{(1+\varepsilon)\sum_{i'} f_{i'}}. \qquad \square$$

Thus, $\boldsymbol{e}_{i^*}^{-1}\sum_j p_{i^*}(j)$ in addition to capturing a *significant* portion of the interval $\sum_{i,j}\boldsymbol{e}_i^{-1}p_i(j)$ is also at least a $1+\varepsilon$ factor larger than the second-largest value. We use these properties to prove the following theorem:

Theorem 13.3.2. *Given that each server j has a non-negative vector $p(j) \in \mathbb{R}^n$, define $q_i \coloneqq \sum_j p_i(j)$. Algorithm 13.1 outputs FAIL only with probability $O(\varepsilon)$ and conditioned on not failing, for all $i \in [n]$*

$$\mathbf{Pr}[\hat{i} = i \textit{ and } \hat{q} \in (1 \pm O(\varepsilon))\frac{q_i}{\sum_i q_i}] = (1 \pm O(\varepsilon))\frac{q_i}{\sum_i q_i} \pm 1/\mathrm{poly}(n).$$

The algorithm uses only one round and has a total communication of $O(s\,\mathrm{polylog}(n)/\varepsilon^2)$ words.

Proof. All the random variables used in the proof are as defined in the algorithm. The algorithm fails to sample if $\max_i \boldsymbol{X}_i \leq S/(2C\log^2 n)$ or if $\boldsymbol{X}_{(1)} \leq (1+\varepsilon/2)\boldsymbol{X}_{(2)}$. Let $\mathscr{E}$ denote the event that $\max_i \boldsymbol{e}_i^{-1}q_i \geq \sum_i \boldsymbol{e}_i^{-1}q_i/4C\log^2 n$ and $(\max_i \boldsymbol{e}_i^{-1}q_i) \geq (1+\varepsilon/4)\cdot \text{second-max}_i\, \boldsymbol{e}_i^{-1}q_i$. We now bound $\mathbf{Pr}[\mathscr{E} \mid \neg\mathrm{FAIL}]$. By Bayes' theorem, we have

$$\mathbf{Pr}[\mathscr{E} \mid \neg\mathrm{FAIL}] = \frac{\mathbf{Pr}[\neg\mathrm{FAIL} \mid \mathscr{E}]\,\mathbf{Pr}[\mathscr{E}]}{\mathbf{Pr}[\neg\mathrm{FAIL} \mid \mathscr{E}]\,\mathbf{Pr}[\mathscr{E}] + \mathbf{Pr}[\neg\mathrm{FAIL} \mid \neg\mathscr{E}]\,\mathbf{Pr}[\neg\mathscr{E}]}.$$

We have $\mathbf{Pr}[\mathscr{E}] \geq 1/(1+\varepsilon/4) - 1/\mathrm{poly}(n)$ from the above lemma. Let $\mathscr{E}'$ denote the event that $\max_i \boldsymbol{e}_i^{-1}q_i \geq \sum_i \boldsymbol{e}_i^{-1}q_i/C\log^2 n$ and $(\max_i \boldsymbol{e}_i^{-1}q_i) \geq (1+\varepsilon)\cdot \text{second-max}_i\, \boldsymbol{e}_i^{-1}q_i$. We note that $\mathscr{E}' \subseteq \mathscr{E}$ and that $\mathbf{Pr}[\mathscr{E}'] \geq 1/(1+\varepsilon) - 1/\mathrm{poly}(n)$. Now,

$$\mathbf{Pr}[\neg\mathrm{FAIL} \mid \mathscr{E}] \geq \mathbf{Pr}[\neg\mathrm{FAIL} \mid \mathscr{E}']\,\mathbf{Pr}[\mathscr{E}' \mid \mathscr{E}] \geq \mathbf{Pr}[\neg\mathrm{FAIL} \mid \mathscr{E}']\,\mathbf{Pr}[\mathscr{E}'].$$

Condition on $\mathscr{E}'$ and let $i^* = \arg\max \boldsymbol{e}_i^{-1}q_i$. By a Chernoff bound, since $S \geq O(C^2\log^5 n/\varepsilon^2)$ we get that with probability $\geq 1 - 1/\mathrm{poly}(n)$

$$\boldsymbol{X}_{i^*} \geq S\frac{\boldsymbol{e}_{i^*}^{-1}q_{i^*}}{\sum_i \boldsymbol{e}_i^{-1}q_i}(1-\varepsilon/5) > \frac{S}{2C\log^2 n}.$$

Additionally, for all other indices i, we get by a Bernstein bound that

$$\mathbf{Pr}[\boldsymbol{X}_i \geq S\frac{\boldsymbol{e}_i^{-1}q_i}{\sum_i \boldsymbol{e}_i^{-1}q_i} + \frac{\varepsilon}{8}\frac{S}{C\log^2 n}] \leq 1/\mathrm{poly}(n).$$

Taking a union bound over all the indices $i \neq i'$, we get that with probability $\geq 1 - 1/\mathrm{poly}(n)$ for all

$i \neq i'$,

$$\boldsymbol{X}_i < S\frac{\boldsymbol{e}_i^{-1}q_i}{\sum_i \boldsymbol{e}_i^{-1}q_i} + \frac{\varepsilon}{4}\frac{S}{C\log^2 n} \leq S\frac{\boldsymbol{e}_{i^*}^{-1}q_{i^*}}{\sum_i \boldsymbol{e}_i^{-1}q_i}(1/(1+\varepsilon)+\varepsilon/8) \leq \frac{1-\varepsilon/5}{1+\varepsilon/2}S\frac{\boldsymbol{e}_{i^*}^{-1}q_{i^*}}{\sum_i \boldsymbol{e}_i^{-1}q_i}$$

when ε is a small enough constant. Hence, $\mathbf{Pr}[\neg\text{FAIL} \mid \mathscr{E}'] \geq 1 - 1/\text{poly}(n)$ and we get

$$\mathbf{Pr}[\neg\text{FAIL} \mid \mathscr{E}] \geq (1 - 1/\text{poly}(n))(1/(1+\varepsilon) - 1/\text{poly}(n)).$$

Now similarly, we show that $\mathbf{Pr}[\text{FAIL} \mid \neg\mathscr{E}]$ is large. Condition on $\neg\mathscr{E}$ and again let $i^* = \max_i \boldsymbol{e}_i^{-1}q_i$. If $\boldsymbol{e}_{i^*}^{-1}q_{i^*}/\sum_i \boldsymbol{e}_i^{-1}q_i \leq 1/4C\log^2 n$, then again by Bernstein's bound, we get $\max_i \boldsymbol{X}_i < S/(2C\log^2 n)$ with probability $\geq 1 - 1/\text{poly}(n)$. Suppose $\boldsymbol{e}_{i^*}^{-1}q_{i^*}/\sum_i \boldsymbol{e}_i^{-1}q_i > 1/4C\log^2 n$ but $\boldsymbol{e}_{i^*}^{-1}q_{i^*}/\sum_i \boldsymbol{e}_i^{-1}q_i < (1+\varepsilon/4) \cdot \text{second-max}_i\, \boldsymbol{e}_i^{-1}q_i$. By using a Chernoff bound, we get $\boldsymbol{X}_{(1)} < (1+\varepsilon/2)\boldsymbol{X}_{(2)}$ with a probability $\geq 1 - 1/\text{poly}(n)$. Hence, $\mathbf{Pr}[\text{FAIL} \mid \neg\mathscr{E}] \geq 1 - 1/\text{poly}(n)$. Thus, overall

$$\mathbf{Pr}[\mathscr{E} \mid \neg\text{FAIL}] = 1 - 1/\text{poly}(n).$$

Now, conditioned on $\neg$FAIL and $\mathscr{E}$, with probability $\geq 1 - 1/\text{poly}(n)$, $\hat{i} = i^*$ and

$$\boldsymbol{Y}_{\hat{i}}/S = \boldsymbol{Y}_{i^*}/S = (1 \pm \varepsilon/2)\frac{\boldsymbol{e}_{i^*}^{-1}q_{i^*}}{\sum_i \boldsymbol{e}_i^{-1}q_i}$$

and hence, $\hat{q} = (1 \pm \varepsilon/2)q_{i^*}/\sum_i q_i$. Finally,

$$\begin{aligned}
\mathbf{Pr}[\hat{i} = i \mid \neg\text{FAIL}] &= \mathbf{Pr}[\hat{i} = i \mid \neg\text{FAIL}, \mathscr{E}]\,\mathbf{Pr}[\mathscr{E} \mid \neg\text{FAIL}] + \mathbf{Pr}[\hat{i} = i \mid \neg\text{FAIL}, \neg\mathscr{E}]\,\mathbf{Pr}[\neg\mathscr{E} \mid \neg\text{FAIL}] \\
&= \mathbf{Pr}[i^* = i \mid \mathscr{E}] \pm 1/\text{poly}(n) \\
&= \frac{q_i}{\sum_i q_i}(1 \pm O(\varepsilon)) \pm 1/\text{poly}(n),
\end{aligned}$$

where the last inequality is from the previous lemma. □

Thus, Algorithm 13.1 can sample approximately from very general "additively-defined" distributions.

13.4 A Two Round Protocol for Sum Approximation

Recall that we say a non-negative function f satisfies "approximate invertibility" with parameters $\theta, \theta', \theta'' > 1$ if the following hold:

1. for all $x_1, x_2 \geq 0$, it holds that $f(x_1) + f(x_2) \leq f(x_1 + x_2)$ which additionally implies that $f(0) = 0$,

Algorithm 13.1: Sampling from General Distributions

```
Input: Each server has a nonnegative vector $p(j) \in \mathbb{R}^n$ and a parameter $\varepsilon$
Output: Samples approximately from the distribution with the probability of $i$ being
        $\sum_j p_i(j) / \sum_{i,j} p_i(j)$
1  All the servers agree on independent exponential random variables $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ using public
     randomness
2  $S \leftarrow O(\varepsilon^{-2} \log^5 n)$
3  For each $j = 1, \ldots, s$, the $j$-th server samples $2S$ independent copies of the random variable
     $\boldsymbol{i}$ defined by $\mathbf{Pr}[\boldsymbol{i} = i] = \boldsymbol{e}_i^{-1} p_i(j) / \sum_i \boldsymbol{e}_i^{-1} p_i(j)$
4  Each server $j$ communicates $\sum_i p_i(j)$, $\sum_i \boldsymbol{e}_i^{-1} p_i(j)$ and the $2S$ sampled coordinates to the
     coordinator
5  The coordinator, using the communication from the servers, samples $2S$ independent
     copies of the random variable $(\boldsymbol{j}, \boldsymbol{i})$ defined by
       $$\mathbf{Pr}[(\boldsymbol{j}, \boldsymbol{i}) = (j, i)] = \frac{\boldsymbol{e}_i^{-1} p_i(j)}{\sum_{i,j} \boldsymbol{e}_i^{-1} p_i(j)}$$
6  $\boldsymbol{X}_i \leftarrow$ The number of times $(*, i)$ is sampled in the first $S$ trials
7  $\boldsymbol{Y}_i \leftarrow$ The number of times $(*, i)$ is sampled in the second $S$ trials
8  $\boldsymbol{X}_{(1)}, \boldsymbol{X}_{(2)} \leftarrow$ top two among $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$
9  if $\boldsymbol{X}_{(1)} < S/(2C \log^2 n)$ or $\boldsymbol{X}_{(1)} \leq (1 + \varepsilon/2)\boldsymbol{X}_{(2)}$ then
10     return FAIL
11 end
12 $\hat{i} \leftarrow \arg\max_i \boldsymbol{Y}_i$
13 $\hat{q} \leftarrow \boldsymbol{e}_{\hat{i}} (\boldsymbol{Y}_{\hat{i}} / S) \sum_{i,j} \boldsymbol{e}_i^{-1} p_i(j) / \sum_{i,j} p_i(j)$
14 return $(\hat{i}, \hat{q})$
```

2. for all x, $f(\theta' x) \geq \theta f(x)$, and
3. for all x, $f(x/(4 \cdot \sqrt{\theta} \cdot \theta')) \geq f(x)/\theta''$.

Note that plugging $x = f^{-1}(z)$ in the second property above, we get $\theta' f^{-1}(z) \geq f^{-1}(\theta z)$. We now define $\varepsilon_1 \coloneqq 1/\theta''$ and $\varepsilon_2 \coloneqq 1 - 1/\theta''$. We can show that for all values of $0 \leq x_2 \leq x_1$, if $f(x_2) \leq \varepsilon_1 f(x_1)$, then $f(x_1 - x_2) \geq (1 - \varepsilon_2) f(x_1)$. This essentially shows that if $f(x_2)$ is very small as compared to $f(x_1)$, then $f(x_1 - x_2)$ can not be very small when compared to $f(x_1)$. These properties make the function f "approximately invertible", meaning that good approximations for $f(x)$ will let us approximate the preimage x as well. We say that a function f that satisfies the above properties is "approximately invertible" with parameters $\theta, \theta', \theta''$.

Note that for an integer $s \geq 1$, we defined $c_f[s]$ to be the smallest number such that for all $x_1, \ldots, x_s \geq 0$,

$$f(x_1 + \cdots + x_s) \leq \frac{c_f[s]}{s} \left(\sqrt{f(x_1)} + \cdots + \sqrt{f(x_s)}\right)^2.$$

Using the Cauchy-Schwarz inequality, we additionally have

$$f(x_1 + \cdots + x_s) \leq c_f[s] \left(f(x_1) + \cdots + f(x_s)\right)$$

for all $x_1, \ldots, x_s \geq 0$. We show that the parameter $c_f[s]$ as a function of s cannot grow arbitrarily. Consider arbitrary integers $s, t \geq 1$. The following lemma shows that $c_f[s \cdot t] \leq c_f[s] \cdot c_f[t]$ which implies that the function $c_f[s]$ is upper bounded by a polynomial in s with a degree that depends only on $c_f[2]$.

Lemma 13.4.1. *For any $x_1, x_2, \ldots, x_{s \cdot t} \geq 0$,*

$$f(x_1 + \cdots + x_{s \cdot t}) \leq \frac{c_f[s] \cdot c_f[t]}{s \cdot t} \left(\sqrt{f(x_1)} + \cdots + \sqrt{f(x_{s \cdot t})}\right)^2.$$

Proof. By definition of $c_f[s]$, we obtain

$$\begin{aligned} &f(x_1 + \cdots + x_{s \cdot t}) \\ &\leq \frac{c_f[s]}{s} \left(\sqrt{f(x_1 + \cdots + x_t)} + \sqrt{f(x_{t+1}) + \cdots + f(x_{2 \cdot t})} + \cdots + \sqrt{f(x_{(s-1) \cdot t+1} + \cdots + f(x_{s \cdot t}))}\right)^2. \end{aligned}$$

Now we use $c_f[t]$ to expand the internal terms, to get

$$f(x_1 + \cdots + x_{s \cdot t}) \leq \frac{c_f[s]}{s} \frac{c_f[t]}{t} \left(\sqrt{f(x_1)} + \cdots + \sqrt{f(x_{s \cdot t})}\right)^2.$$

By definition of $c_f[s \cdot t]$, we obtain $c_f[s \cdot t] \leq c_f[s] \cdot c_f[t]$. □

The above lemma upper bounds the growth of the parameter $c_f[s]$. We now lower bound the growth and show that $c_f[s]$ must grow at least linearly in the parameter s.

Lemma 13.4.2. *If $s \geq t$, then $c_f[s] \geq c_f[t] \cdot s/t$.*

Proof. Let $x_1, \ldots, x_t \geq 0$ be arbitrary.

$$\begin{aligned} f(x_1 + \cdots + x_t) &= f(x_1 + \cdots + x_t + \underbrace{0 + \cdots + 0}_{s-t}) \\ &\leq \frac{c_f[s]}{s}\left(\sqrt{f(x_1)} + \cdots + \sqrt{f(x_t)} + \underbrace{\sqrt{f(0)} + \cdots + \sqrt{f(0)}}_{s-t}\right)^2 \\ &\leq \frac{c_f[s]}{s}(\sqrt{f(x_1)} + \cdots + \sqrt{f(x_t)})^2 \end{aligned}$$

where we used $f(0) = 0$ in the last inequality. Hence, by definition of $c_f[t]$, we get $c_f[t]/t \leq c_f[s]/s$ from which we obtain that $c_f[s] \geq c_f[t] \cdot s/t$. $\square$

Using the above properties, we obtain that $c_f[s] \leq c_f[2^{\lceil \log_2(s) \rceil}] \leq (c_f[2])^{\lceil \log_2(s) \rceil} \leq c_f[2] \cdot s^{\log_2 c_f[2]}$ which shows that c_f only grows at most *polynomially* in the number of servers s and the degree of growth is upper bounded $\log_2 c_f[2]$. For example, if $f(x) = x^k$, then we can show $c_f[2] = 2^{k-1}$ from which we obtain that $c_f[s] \leq (2s)^{k-1}$ for all the values of s.

In addition, if $c_f[t] \geq c_f[s]/\alpha$ for some value α, using the second property above, we obtain $c_f[s] \geq c_f[t] \cdot s/t \geq (c_f[s]/\alpha) \cdot (s/t)$ which implies $t \geq s/\alpha$. Thus, if $c_f[t]$ is "comparable" to $c_f[s]$, then t is "comparable" to s as well. This is a property that we critically use in the analysis of our algorithm.

Protocol Description and Analysis

We now recall the overview of the algorithm we presented in the introduction and define some notation that we use throughout our analysis. All the servers together with the coordinator use the shared randomness and sample standard exponential random variables $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ and the goal of the coordinator is to compute

$$\max_i \boldsymbol{e}_i^{-1} f(x_i).$$

Define $i^* \coloneqq \arg\max_i \boldsymbol{e}_i^{-1} f(x_i)$. We have seen that the median of $O(1/\varepsilon^2)$ independent copies of the random variable $\max_i \boldsymbol{e}_i^{-1} f(x_i)$ can be used to compute a $1 \pm \varepsilon$ approximation to $\sum_i f(x_i)$ with high probability. Throughout the analysis, we condition on the event that

$$\sum_i \boldsymbol{e}_i^{-1} f(x_i) \leq (C \log^2 n) \cdot \max_i \boldsymbol{e}_i^{-1} f(x_i).$$

We have seen that this event holds with probability $\geq 1 - 1/\text{poly}(n)$ over the exponential random variables. Now fix a server $j \in [s]$. The server j samples N independent copies of the random variable $\boldsymbol{i}$ supported in the set $[n]$ and has a distribution defined as

$$\mathbf{Pr}[\boldsymbol{i} = i] = \frac{\boldsymbol{e}_i^{-1} f(x_i)}{\sum_i \boldsymbol{e}_i^{-1} f(x_i)}.$$

Let SC_j denote the set of coordinates that are sampled by the server j. The server j then sends the set SC_j along with (i) the sum $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$ and (ii) the values $x_i(j)$ for $i \in \mathsf{SC}_j$ to the coordinator using $O(N)$ words of communication.

The coordinator defines the set $\mathsf{SC} \coloneqq \bigcup_j \mathsf{SC}_j$ as the union of the sets of coordinates that are sampled at different servers. We will first show that the coordinate $i^* \in \mathsf{SC}$ with a large probability if N is large enough.

13.4.1 The Coordinate i^* is Sampled

Lemma 13.4.3. *If $N \geq (C_{13.4.3} \log^3 n) \cdot c_f[s]/s$ for a universal constant $C_{13.4.3}$ large enough, then the coordinate $i^* \in \mathsf{SC}$ with a probability $\geq 1 - 1/n^4$.*

Proof. Let $q_{i^*}(j) = \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}(j))/\sum_{i \in [n]} \boldsymbol{e}_i^{-1} f(x_i(j))$ be the probability that the random variable $\boldsymbol{i} = i^*$ (note that the random variable $\boldsymbol{i}$ has a different distribution at each server) and $p_{i^*}(j)$ be the probability that $i^* \in \mathsf{SC}_j$. We have

$$p_{i^*}(j) = 1 - (1 - q_{i^*}(j))^N.$$

If $q_{i^*}(j) = (4 \log n)/N$, then $p_{i^*}(j) \geq 1-(1-q_{i^*}(j))^N \geq 1-\exp(-q_{i^*}(j)N) \geq 1-\exp(-4\log n) \geq 1-1/n^4$ which implies that $i^* \in \mathsf{SC}_j \subseteq \mathsf{SC}$ with probability $\geq 1-1/n^4$, and we are done. Otherwise,

$$p_{i^*}(j) = 1 - (1 - q_{i^*}(j))^N \geq \frac{N q_{i^*}(j)}{4 \log n}(1 - 1/n^4) \tag{13.7}$$

using the concavity of the function $1 - (1-x)^N$ in the interval $[0, 4\log n/N]$.

Since each of the servers samples the coordinates independently, the probability that $i^* \in \mathsf{SC}$ is $1 - (1 - p_{i^*}(1))(1 - p_{i^*}(2)) \cdots (1 - p_{i^*}(s)) \geq 1 - \exp(-\sum_j p_{i^*}(j))$. So, showing that the sum $\sum_j p_{i^*}(j)$ is large implies that i^* is in the set SC with a large probability. Assume that for all j, $q_{i^*}(j) < 4\log n/N$ since otherwise we already have that the coordinate $i^* \in \mathsf{SC}$ with probability $\geq 1-1/n^4$. Now using (13.7)

$$\sum_j p_{i^*}(j) \geq \frac{N}{8 \log n} \sum_j q_{i^*}(j) = \frac{N}{8 \log n} \sum_j \frac{\boldsymbol{e}_{i^*}^{-1} f(x_{i^*}(j))}{\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))}.$$

We will now give a lower bound on the sum in the above equation using the following simple lemma and the definition of the parameter $c_f[s]$.

Lemma 13.4.4. *Let $a_1, \dots, a_s \geq 0$ and $b_1, \dots, b_s > 0$ be arbitrary. Then,*

$$\sum_{j \in [s]} \frac{a_j}{b_j} \geq \frac{(\sum_{j \in [s]} \sqrt{a_j})^2}{\sum_j b_j}.$$

Proof. We prove the lemma by using the Cauchy-Schwarz inequality. Since we assume that $b_j > 0$ for all j, we can write

$$\sum_{j \in [s]} \sqrt{a_j} = \sum_{j \in [s]} \sqrt{\frac{a_j}{b_j}} \sqrt{b_j}.$$

Using the Cauchy-Schwarz inequality, we get

$$\sum_{j \in [s]} \sqrt{a_j} = \sum_{j \in [s]} \sqrt{\frac{a_j}{b_j}} \sqrt{b_j} \leq \sqrt{\sum_{j \in [s]} \frac{a_j}{b_j}} \sqrt{\sum_{j \in [s]} b_j}.$$

Squaring both sides and using the fact that $\sum_{j \in [s]} b_j > 0$, we get

$$\sum_{j \in [s]} \frac{a_j}{b_j} \geq \frac{(\sum_{j \in [s]} \sqrt{a_j})^2}{\sum_{j \in [s]} b_j}. \qquad \square$$

Letting $a_j = \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}(j))$ and $b_j = \sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$ in the above lemma, we get

$$\sum_j \frac{\boldsymbol{e}_{i^*}^{-1} f(x_{i^*}(j))}{\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))} \geq \frac{\boldsymbol{e}_{i^*}^{-1} (\sum_{j \in [s]} \sqrt{f(x_{i^*}(j))})^2}{\sum_j \sum_i \boldsymbol{e}_i^{-1} f(x_i(j))}.$$

Using the definition of $c_f[s]$, we obtain

$$\left(\sum_{j \in [s]} \sqrt{f(x_{i^*}(j))} \right)^2 \geq \frac{s}{c_f[s]} (f(x_{i^*}(1) + \cdots x_{i^*}(s))) = \frac{s}{c_f[s]} f(x_{i^*}).$$

Using the super-additivity of the function f, we get

$$\sum_j \sum_i \boldsymbol{e}_i^{-1} f(x_i(j)) = \sum_i \boldsymbol{e}_i^{-1} \sum_j f(x_i(j)) \leq \sum_i \boldsymbol{e}_i^{-1} f(\sum_j x_i(j)) = \sum_i \boldsymbol{e}_i^{-1} f(x_i).$$

Since we conditioned on the event that $\sum_i \boldsymbol{e}_i^{-1} f(x_i) \le (C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$, we get

$$\sum_j \sum_i \boldsymbol{e}_i^{-1} f(x_i(j)) \le (C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}).$$

We therefore have

$$\sum_j \frac{\boldsymbol{e}_{i^*}^{-1} f(x_{i^*}(j))}{\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))} \ge \frac{s}{c_f[s]} \frac{\boldsymbol{e}_{i^*}^{-1} f(x_{i^*})}{(C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})} \ge \frac{s}{C \log^2 n \cdot c_f[s]}.$$

Thus, if $N \ge (32C \log^3 n) \cdot c_f[s]/s$, then $\sum_j p_{i^*}(j) \ge 4 \log n$ which implies that i^* is in the set **SC** with probability $\ge 1 - 1/n^4$. Letting $C_{13.4.3} \coloneqq 32C$, we have the proof. ◻

When $N \gg c_f[s] \log^3 n/s$ as is required by the above lemma, the set $\mathbf{SC} = \bigcup_j \mathbf{SC}_j$ may have a size of $\Omega(c_f[s] \log^3 n)$ which is quite large. Conditioned on the event that $i^* \in \mathbf{SC}$, we now want to compute a small subset $\mathbf{PL} \subseteq \mathbf{SC}$ such that $i^* \in \mathbf{PL}$ with a large probability.

13.4.2 Computing the set PL

Recall we condition on the event that

$$\sum_i \boldsymbol{e}_i^{-1} f(x_i) \le (C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$$

and that $i^* \in \mathbf{SC}$. As we noted in the introduction, we proceed by constructing an estimator $\hat{x}_i$ for each $i \in \mathbf{SC}$ that satisfies the following properties with a probability $1 - 1/\text{poly}(n)$:

1. For all $i \in \mathbf{SC}$, $\hat{x}_i \le x_i$ and
2. $\boldsymbol{e}_{i^*}^{-1} f(\hat{x}_{i^*}) \ge \alpha \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$ for some $\alpha < 1$.

Fix an index $i \in [n]$. In the remaining part of this section, we will describe a quantity that the coordinator can approximate to obtain $\hat{x}_i$ which satisfies the above properties.

Contribution from Large Servers

We define $\textsc{Large}_i$ to be the set of servers j such that

$$q_i(j) = \frac{\boldsymbol{e}_i^{-1} f(x_i(j))}{\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))} \ge \frac{4 \log n}{(c_f[s] \log^3 n)/s}.$$

Note if $i \in \mathbf{SC}_j$, then the coordinator can determine if $j \in \textsc{Large}_i$ since it has access to both the values $\boldsymbol{e}_i^{-1} f(x_i(j))$ and $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$. We have the following lemma:

Lemma 13.4.5. *If $N \ge (c_f[s]\log^3 n)/s$, then with probability $\ge 1 - 1/n^3$, for all the coordinates i and servers $j \in \text{LARGE}_i$, we have $i \in \mathbf{SC}_j$. In other words, with a large probability, all the coordinates i are sampled at all the servers that are in the set LARGE_i.*

Proof. Recall that $\mathbf{SC}_j$ denotes the set of coordinates that are sampled at server j. Consider a fixed $i \in [n]$. If $j \in \text{LARGE}_i$, then the probability that i is sampled at server j is $1-(1-q_i(j))^N \ge 1-(1-4\log n/(c_f[s]\log^3 n/s))^N \ge 1-\exp(-4\log n) \ge 1-1/n^4$ using $N \ge (c_f[s]\log^3 n)/s$. By a union bound over all the servers j that are LARGE for i, we have the proof. □

Thus, with a large probability, for each $i \in [n]$, the contribution to $x_i = \sum_j x_i(j)$ from servers $j \in \text{LARGE}_i$ can be computed exactly by the coordinator after it receives the samples from the servers.

Contribution from SMALL Servers

We now define SMALL_i to be the set of servers j for which

$$q_i(j) = \frac{\boldsymbol{e}_i^{-1} f(x_i(j))}{\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))} \le \frac{\varepsilon_1}{c_f[s]\cdot(C\log^2 n)}.$$

We have

$$\sum_{j\in\text{SMALL}_i} \boldsymbol{e}_i^{-1} f(x_i(j)) \le \sum_{j\in\text{SMALL}_i} \frac{\varepsilon_1 \sum_{i'} \boldsymbol{e}_{i'}^{-1} f(x_{i'}(j))}{c_f[s]\cdot(C\log^2 n)} \le \frac{\varepsilon_1\cdot(C\log^2 n)\cdot(\boldsymbol{e}_{i^*}^{-1} f(x_{i^*}))}{c_f[s]\cdot(C\log^2 n)} = \frac{\varepsilon_1\cdot(\boldsymbol{e}_{i^*}^{-1} f(x_{i^*}))}{c_f[s]}$$

where we used the fact that

$$\sum_{j\in\text{SMALL}_i}\sum_{i'} \boldsymbol{e}_{i'}^{-1} f(x_{i'}(j)) = \sum_{i'}\sum_{j\in\text{SMALL}_i} \boldsymbol{e}_{i'}^{-1} f(x_{i'}(j)) \le \sum_{i'} \boldsymbol{e}_{i'}^{-1} f(x_{i'}) \le (C\log^2 n)\cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}).$$

By definition of the parameter $c_f[s]$, we then obtain that

$$\boldsymbol{e}_i^{-1} f(\sum_{j\in\text{SMALL}_i} x_i(j)) \le c_f[s]\cdot \boldsymbol{e}_i^{-1} \sum_{j\in\text{SMALL}_i} f(x_i(j)) \le \varepsilon_1\cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$$

which then implies

$$\boldsymbol{e}_{i^*}^{-1} f(x_{i^*} - \sum_{j\in\text{SMALL}_{i^*}} x_{i^*}(j)) \ge (1-\varepsilon_2)\cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}). \tag{13.8}$$

Hence we can ignore the contribution of the servers in SMALL_i when computing $\hat{x}_i$.

Contribution from Remaining Servers

From the above, we have for each coordinate $i \in [n]$, we can compute the contribution of servers $j \in \textsc{Large}_i$ exactly and ignore the contribution of servers in $\textsc{Small}_i$ while still being able to satisfy the required properties for $\hat{x}_i$. We will now show how to estimate the contribution of servers j that are neither in $\textsc{Large}_i$ nor in $\textsc{Small}_i$. Note that for such servers, the value $q_i(j) = \boldsymbol{e}_i^{-1} f(x_i(j)) / \sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$ lies in the interval

$$\left[\frac{\varepsilon_1}{c_f[s] \cdot (C \log^2 n)}, \frac{4s}{c_f[s] \log^2 n}\right].$$

We now partition[4] the above interval into intervals with lengths geometrically increasing by a factor of $\sqrt{\theta}$. Let $P_{\text{start}} = \frac{\varepsilon_1}{c_f[s] \cdot (C \log^2 n)}$ and we partition the above interval into intervals $[P_{\text{start}}, \sqrt{\theta} P_{\text{start}}]$, $[\sqrt{\theta} P_{\text{start}}, (\sqrt{\theta})^2 P_{\text{start}}], \ldots$, and so on. We note that there are at most

$$A = O\left(\frac{\log(s/\varepsilon_1)}{\log \theta}\right) \tag{13.9}$$

such intervals in the partition.

Let $I_i^{(a)}$ denote the set of servers j such that $q_i(j) \in [(\sqrt{\theta})^a P_{\text{start}}, (\sqrt{\theta})^{a+1} P_{\text{start}}]$. If $|I_i^{(a)}|$ is large enough, then the number of servers j in $I_i^{(a)}$ at which the coordinate i is sampled is "concentrated" which can then be used to estimate $|I_i^{(a)}|$. But observe that even having an estimate of $|I_i^{(a)}|$ is insufficient since we cannot directly estimate $\sum_{j \in I^{(a)}} x_i(j)$ only given $|I_i^{(a)}|$ as the servers in the set $I_i^{(a)}$ may have quite different values for $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$. So we further partition the servers based on the value of $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$. We first give a lower bound on the values $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$ that we need to consider.

Lemma 13.4.6. *If the server $j \notin \textsc{Small}_i \cup \textsc{Large}_i$ and $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j)) \le \frac{\varepsilon_1(1-\varepsilon_2) \sum_{i,j} \boldsymbol{e}_i^{-1} f(x_i(j))}{4Cs^2}$, then the contribution from all those servers can be ignored.*

Proof. Let $\textsc{Ignore}_i$ be the set of servers j that are not in $\textsc{Large}_i \cup \textsc{Small}_i$ and have $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j)) \le \frac{\sum_{i,j} \varepsilon_1(1-\varepsilon_2) \boldsymbol{e}_i^{-1} f(x_i(j))}{4Cs^2}$. By definition, we have

$$\begin{aligned}
\sum_{j \in \textsc{Ignore}_i} \boldsymbol{e}_i^{-1} f(x_i(j)) &\le \sum_{j \in \textsc{Ignore}_i} \frac{4s}{c_f[s] \log^2 n} \sum_i \boldsymbol{e}_i^{-1} f(x_i(j)) \\
&\le \frac{4s \cdot |\textsc{Ignore}_i|}{c_f[s] \log^2 n} \frac{\varepsilon_1(1-\varepsilon_2) \sum_{i,j} \boldsymbol{e}_i^{-1} f(x_i(j))}{4Cs^2}.
\end{aligned}$$

[4]We do not require it and so are not too careful about ensuring that the intervals we use are disjoint.

Using the definition of $c_f[s]$ and the Cauchy-Schwarz inequality, we then obtain

$$\begin{aligned}\boldsymbol{e}_i^{-1} f(\sum_{j\in\textsc{Ignore}_i} x_i(j)) &\le \frac{c_f[s]}{s}\cdot|\textsc{Ignore}_i|\cdot\sum_{j\in\textsc{Ignore}_i}\boldsymbol{e}_i^{-1}f(x_i(j))\\ &\le \frac{4|\textsc{Ignore}_i|^2}{\log^2 n}\frac{\varepsilon_1(1-\varepsilon_2)\sum_{i,j}\boldsymbol{e}_i^{-1}f(x_i(j))}{4Cs^2}.\end{aligned}$$

Since $\sum_{i,j}\boldsymbol{e}_i^{-1}f(x_i(j)) \le (C\log^2 n)\cdot\boldsymbol{e}_{i^*}^{-1}f(x_i^*)$ and $|\textsc{Ignore}_i|\le s$, we get

$$\boldsymbol{e}_i^{-1}f(\sum_{j\in\textsc{Ignore}_i} x_i(j)) \le \varepsilon_1(1-\varepsilon_2)\boldsymbol{e}_{i^*}^{-1}f(x_i^*).$$

Taking $i=i^*$, we get $\boldsymbol{e}_{i^*}^{-1}f(\sum_{j\in\textsc{Ignore}_{i^*}} x_{i^*}(j)) \le \varepsilon_1(1-\varepsilon_2)\boldsymbol{e}_{i^*}^{-1}f(x_{i^*})$ and therefore using (13.8) we obtain that

$$\boldsymbol{e}_{i^*}^{-1}f(\sum_{j\in\textsc{Ignore}_{i^*}} x_{i^*}(j)) \le \boldsymbol{e}_{i^*}^{-1}\varepsilon_1 f(x_{i^*}-\sum_{j\in\textsc{Small}_{i^*}} x_{i^*}(j))$$

from which we then get

$$\boldsymbol{e}_{i^*}^{-1}f(x_{i^*}-\sum_{j\in\textsc{Small}_{i^*}} x_{i^*}(j)-\sum_{j\in\textsc{Ignore}_{i^*}} x_{i^*}(j)) \ge (1-\varepsilon_2)^2\boldsymbol{e}_{i^*}^{-1}f(x_{i^*}).$$

Thus, the contribution from the servers in the set $\textsc{Ignore}_{i^*}$ can be ignored as we can make $\boldsymbol{e}_{i^*}^{-1}f(\hat{x}_{i^*})$ large even without them. □

So we only have to focus on the servers for which the quantity $\sum_i \boldsymbol{e}_i^{-1}f(x_i(j))$ lies in the interval

$$\left[\frac{\varepsilon_1(1-\varepsilon_2)}{4Cs^2}\sum_{i,j}\boldsymbol{e}_i^{-1}f(x_i(j)), \max_j \sum_i \boldsymbol{e}_i^{-1}f(x_i(j))\right].$$

Now define $F_{\text{start}} \coloneqq \frac{\varepsilon_1(1-\varepsilon_2)}{4Cs^2}\sum_{i,j}\boldsymbol{e}_i^{-1}f(x_i(j))$ and split the above interval into intervals $[F_{\text{start}}, (\sqrt{\theta}F_{\text{start}})]$, $[(\sqrt{\theta})F_{\text{start}}, (\sqrt{\theta})^2F_{\text{start}}], \ldots$, and so on. Note that there are at most

$$B = O\left(\frac{\log(s^2/\varepsilon_1(1-\varepsilon_2))}{\log\theta}\right) \tag{13.10}$$

such intervals. Let $I_i^{(a,b)}$ for $a=0,1,\ldots,A-1$ and $b=0,1,\ldots,B-1$ be the set of servers j for which

$$\frac{\boldsymbol{e}_i^{-1}f(x_i(j))}{\sum_i \boldsymbol{e}_i^{-1}f(x_i(j))} \in [(\sqrt{\theta})^a P_{\text{start}}, (\sqrt{\theta})^{a+1}P_{\text{start}}],$$

and

$$\sum_i \boldsymbol{e}_i^{-1} f(x_i(j)) \in [(\sqrt{\theta})^b F_{\text{start}}, (\sqrt{\theta})^{b+1} F_{\text{start}}].$$

If the set $|I_i^{(a,b)}|$ is large, then the number of servers $j \in I_i^{(a,b)}$ at which i is sampled is concentrated which can then in turn be used to approximate $|I_i^{(a,b)}|$. But if $|I_i^{(a,b)}|$ is too small, then we cannot obtain a good approximation using the number of servers in the set $I_i^{(a,b)}$ that sample i. In the following lemma, we show that the contribution from servers in the sets $I_i^{(a,b)}$ needs to be considered only if $|I_i^{(a,b)}|$ is large. Let BAD_i denote the set of tuples (a, b) for which

$$c_f[|I_i^{(a,b)}|] \leq \frac{c_f[s]}{c_f[A \cdot B] \cdot (A \cdot B) \cdot (\sqrt{\theta})^{a+1}/(1-\varepsilon_2)^2}. \tag{13.11}$$

Let GOOD_i be the set of all the remaining tuples (a, b). We first note that the coordinator cannot determine if a particular tuple (a, b) is in the set BAD_i. Using the properties of $c_f[s]$, we get that if $(a, b) \in \text{GOOD}_i$, then

$$|I_i^{(a,b)}| \geq \frac{s}{c_f[A \cdot B] \cdot (A \cdot B) \cdot (\sqrt{\theta})^{a+1}/(1-\varepsilon_2)^2}.$$

Lemma 13.4.7. *The contribution from servers $j \in \bigcup_{(a,b)\in\text{BAD}_i} I_i^{(a,b)}$ can be ignored.*

Proof. By definition of the set of servers $I_i^{(a,b)}$,

$$\begin{aligned}
\sum_{j\in I_i^{(a,b)}} \boldsymbol{e}_i^{-1} f(x_i(j)) &\leq \sum_{j\in I_i^{(a,b)}} (\sqrt{\theta})^{a+1} P_{\text{start}} \sum_{i'\in[n]} \boldsymbol{e}_{i'}^{-1} f(x_{i'}(j)) \\
&\leq (\sqrt{\theta})^{a+1} P_{\text{start}} \sum_{i'\in[n]} \boldsymbol{e}_{i'}^{-1} \sum_{j\in I_i^{(a,b)}} f(x_{i'}(j)) \\
&\leq (\sqrt{\theta})^{a+1} P_{\text{start}} \sum_{i'\in[n]} \boldsymbol{e}_{i'}^{-1} f(x_{i'}) \\
&\leq (\sqrt{\theta})^{a+1} P_{\text{start}} \cdot (C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}).
\end{aligned}$$

By definition of the parameter c_f, we have

$$\boldsymbol{e}_i^{-1} f\left(\sum_{(a,b)\in\text{BAD}_i} \sum_{j\in I_i^{(a,b)}} x_i(j) \right)$$

$$\begin{aligned}
&\leq c_f[A \cdot B]\left(\sum_{(a,b)\in\text{BAD}_i} \boldsymbol{e}_i^{-1} f\big(\sum_{j\in I_i^{(a,b)}} x_i(j)\big)\right)\\
&\leq c_f[A \cdot B] \sum_{(a,b)\in\text{BAD}_i} c_f[|I_i^{(a,b)}|] \sum_{j\in I_i^{(a,b)}} \boldsymbol{e}_i^{-1} f(x_i(j))\\
&\leq c_f[A \cdot B] \sum_{(a,b)\in\text{BAD}_i} c_f[|I_i^{(a,b)}|] \cdot (\sqrt{\theta})^{a+1} P_{\text{start}} \cdot (C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})\\
&\leq c_f[A \cdot B] \sum_{(a,b)\in\text{BAD}_i} c_f[|I_i^{(a,b)}|] \cdot (\sqrt{\theta})^{a+1} \cdot \frac{\varepsilon_1}{c_f[s] \cdot (C \log^2 n)} \cdot (C \log^2 n) \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}).
\end{aligned}$$

Using (13.11), we get

$$\boldsymbol{e}_i^{-1} f\left(\sum_{(a,b)\in\text{BAD}_i} \sum_{j\in I_i^{(a,b)}} x_i(j)\right) \leq (1-\varepsilon_2)^2 \varepsilon_1 \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}).$$

Taking $i = i^*$, we get that

$$\boldsymbol{e}_{i^*}^{-1} f\big(x_{i^*} - \sum_{j\in\text{SMALL}_{i^*}} x_{i^*}(j) - \sum_{j\in\text{IGNORE}_{i^*}} x_{i^*}(j) - \sum_{(a,b)\in\text{BAD}_i} \sum_{j\in I_{i^*}^{(a,b)}} x_{i^*}(j)\big) \geq (1-\varepsilon_2)^3 \cdot \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}).$$

Thus the contribution from the servers in the set $I_i^{(a,b)}$ for tuples $(a, b) \in \text{BAD}_i$ can be ignored. □

Now we show that the coordinator can compute $\hat{x}_i$ by essentially approximating the following quantity:

$$\begin{aligned}
&x_i - \sum_{j\in\text{SMALL}_i} x_i(j) - \sum_{j\in\text{IGNORE}_i} x_i(j) - \sum_{(a,b)\in\text{BAD}_i} \sum_{j\in I_i^{(a,b)}} x_i(j)\\
&= \sum_{j\in\text{LARGE}_i} x_i(j) + \sum_{(a,b)\in\text{GOOD}_i} \sum_{j\in I_i^{(a,b)}} x_i(j).
\end{aligned}$$

Algorithm to compute $\hat{x}_i$

We will go on to give an algorithm that can approximate $\hat{x}_i$ given the samples that the coordinator receives in the first round. We have already seen that given an index i, all the servers in the set LARGE_i sample the coordinate i with a large probability and therefore we can compute the quantity $\sum_{j\in\text{LARGE}_i} x_i(j)$ exactly with high probability. We have also seen that the contribution from servers that are in the set SMALL_i and in the set IGNORE_i can be ignored.

The main remaining contribution to x_i that is to be accounted is from tuples $(a, b) \in \text{GOOD}_i$. An important issue we need to solve for is the fact that the coordinator cannot determine if a given tuple (a, b) is in the set $\textsc{Good}_i$ or in $\textsc{Bad}_i$. We will show that if $(a, b) \in \textsc{Good}_i$, then i is sampled at many servers in the set $I_i^{(a,b)}$ and depending on the *absolute* number of servers in $I_i^{(a,b)}$ that sample i, we mark a tuple (a, b) as "probably good for i". We argue that, with high probability, all tuples $(a, b) \in \textsc{Good}_i$ are marked "probably good for i" and that for the tuples (a, b) in $\textsc{Bad}_i$ that are marked "probably good for i", we will still obtain good approximations for $|I_i^{(a,b)}|$.

The following lemma shows why approximating $|I_i^{(a,b)}|$ is enough to approximate the contributions of the servers in the set $I_i^{(a,b)}$.

Lemma 13.4.8. *The value $|I_i^{(a,b)}|$ can be used to approximate $\sum_{j \in I_i^{(a,b)}} x_i(j)$ up to a factor of θ'.*

Proof. For all servers $j \in I_i^{(a,b)}$, by definition, we have

$$(\sqrt{\theta})^{a+b} P_{\text{start}} F_{\text{start}} \le \boldsymbol{e}_i^{-1}(f(x_i(j))) \le (\sqrt{\theta})^{a+b+2} P_{\text{start}} F_{\text{start}}$$

which further implies using the monotonicity of f that

$$f^{-1}(\boldsymbol{e}_i(\sqrt{\theta})^{a+b} P_{\text{start}} F_{\text{start}}) \le x_i(j) \le f^{-1}(\boldsymbol{e}_i(\sqrt{\theta})^{a+b+2} P_{\text{start}} F_{\text{start}}) \le \theta' f^{-1}(\boldsymbol{e}_i(\sqrt{\theta})^{a+b} P_{\text{start}} F_{\text{start}}).$$

Now we note

$$\frac{\sum_{j \in I_i^{(a,b)}} x_i(j)}{\theta'} \le |I_i^{(a,b)}| \cdot f^{-1}(\boldsymbol{e}_i(\sqrt{\theta})^{a+b} P_{\text{start}} F_{\text{start}}) \le \sum_{j \in I_i^{(a,b)}} x_i(j). \qquad \square$$

We will then show that the expected number of servers in the set $I_i^{(a,b)}$ that sample i can be used to obtain an approximation for $|I_i^{(a,b)}|$.

Lemma 13.4.9. *Let $p_i(j) \coloneqq 1 - (1 - q_i(j))^N$ denote the probability that the coordinate i is among the N coordinates sampled at server j. For any tuple (a, b),*

$$|I_i^{(a,b)}| \cdot \frac{(1 - (1 - (\sqrt{\theta})^{a+1} P_{start})^N)}{\sqrt{\theta}} \le \sum_{j \in I_i^{(a,b)}} p_i(j) \le |I_i^{(a,b)}| \cdot (1 - (1 - (\sqrt{\theta})^{a+1} P_{start})^N).$$

Proof. Using monotonicity and concavity of the function $1 - (1 - x)^N$ in the interval $[0, 1]$, for all $j \in I_i^{(a,b)}$, we have

$$(1 - (1 - (\sqrt{\theta})^{a+1} P_{\text{start}})^N) \ge p_i(j) \ge \frac{(1 - (1 - (\sqrt{\theta})^{a+1} P_{\text{start}})^N)}{\sqrt{\theta}}.$$

Summing the inequalities over all $j \in I_i^{(a,b)}$ gives the proof. $\square$

If $\sum_{j \in I_i^{(a,b)}} p_j$ is large enough, we obtain using a Chernoff bound that the number of servers in $I_i^{(a,b)}$ at which i is sampled is highly concentrated around the mean $\sum_{j \in I_i^{(a,b)}} p_j$ and using the above lemmas, we can obtain an estimate for $|I_i^{(a,b)}|$ and therefore estimate $\sum_{j \in I_i^{(a,b)}} x_i(j)$. We will follow this approach to approximate $\sum_{j \in I_i^{(a,b)}} x_i(j)$. Let $X_i^{(a,b)}$ be the number of servers in $j \in I_i^{(a,b)}$ that sample the coordinate i. By linearity of expectation, we have $\mathbf{E}[X_i^{(a,b)}] = \sum_{j \in I_i^{(a,b)}} p_j$. We will now use the following standard concentration bounds.

Lemma 13.4.10. *Let Y_j for $j \in [t]$ be a Bernoulli random variable with $\mathbf{Pr}[Y_j = 1] = p_j$. Let $Y_1, \dots, Y_t$ be mutually independent and $X = Y_1 + \cdots + Y_t$. Then the following inequalities hold:*

1. *If $\sum_j p_j \geq 100 \log n$, then*
$$\mathbf{Pr}[X = (1 \pm 1/3) \sum_{j=1}^{t} p_j] \geq 1 - 1/n^3.$$
2. *For any values of $p_1, \dots, p_t$,*
$$\mathbf{Pr}[X < 2 \sum_{j=1}^{t} p_j + 4 \log n] \geq 1 - 1/n^4.$$

Proof. To prove the first inequality, we use the multiplicative Chernoff bound. We have
$$\mathbf{Pr}[X = (1 \pm 1/3) \sum_{j=1}^{t} p_j] \leq 2 \exp\left(-\frac{\sum_{j=1}^{t} p_j}{27}\right).$$
If $\sum_{j=1}^{t} p_j \geq 100 \log n$, then the RHS is at most $1/n^3$. To prove the second inequality, we use the Bernstein concentration bound. We get
$$\mathbf{Pr}[X \geq 2 \sum_{j=1}^{t} p_j + 4 \log n] \leq \exp\left(-\frac{(\sum_j p_j + 4 \log n)^2}{\sum_j p_j + (\sum_j p_j + 4 \log n)/3}\right) \leq \exp(-4 \log n) \leq 1/n^4. \quad \square$$

We note that if $\sum_j p_j \geq 100 \log n$, then with probability $\geq 1 - 1/n^3$, we have $X = (1 \pm 1/3) \log n \geq 66 \log n$ and if $\sum_j p_j \leq 30 \log n$, then with probability $\geq 1 - 1/n^4$, we have $X < 64 \log n$. Thus the value of X can be used to separate the cases of $\sum_j p_j \geq 100 \log n$ or $\sum_j p_j \leq 30 \log n$ with high probability.

Now we show that if $(a, b) \in \textsc{Good}_i$ and N is large enough, then $\sum_{j \in I_i^{(a,b)}} p_j \geq 100 \log n$.

Lemma 13.4.11. *If $(a,b) \in \textsc{Good}_i$ and the number of samples N at each coordinator satisfies*

$$N \geq O\left(\frac{c_f[s]}{s} \cdot \frac{c_f[A \cdot B] \cdot (A \cdot B) \cdot \sqrt{\theta} \cdot \log^4 n}{\varepsilon_1 (1-\varepsilon_2)^2}\right),$$

then either $(\sqrt{\theta})^{a+1} P_{start} \geq 4 \log n / N$ or $\sum_{j \in I_i^{(a,b)}} p_j \geq 100 \log n$.

Proof. Assume $(\sqrt{\theta})^{a+1} P_{\text{start}} < 4 \log n / N$. By concavity of the function $1 - (1-x)^N$ in the interval $[0, 1]$,

$$1 - (1 - (\sqrt{\theta})^{a+1} P_{\text{start}})^N \geq \frac{N (\sqrt{\theta})^{a+1} P_{\text{start}}}{4 \log n} (1 - 1/n^4).$$

For $(a,b) \in \textsc{Good}_i$, we then obtain

$$\begin{aligned}\sum_{j \in I_i^{(a,b)}} p_j &\geq |I_i^{(a,b)}| \cdot \frac{N (\sqrt{\theta})^{a+1} P_{\text{start}}}{4 \log n \sqrt{\theta}} (1 - 1/n^4) \\ &\geq \frac{s}{c_f[A \cdot B] \cdot (A \cdot B) \cdot (\sqrt{\theta})^{a+1} / (1-\varepsilon_2)^2} \frac{N (\sqrt{\theta})^{a+1} \varepsilon_1}{4 C \log^3 n \cdot c_f[s] \cdot \sqrt{\theta}} \cdot (1 - 1/n^4) \\ &\geq 100 \log n.\end{aligned}$$

□

With a high probability, for all coordinates i and all servers j in the set $I_i^{(a,b)}$ for some a with $(\sqrt{\theta})^{a+1} P_{\text{start}} \geq 4 \log n / N$, the coordinate $i \in \mathbf{SC}_j$. So the contribution from such servers can be computed exactly akin to the servers in the set $\textsc{Large}_j$.

Now consider all the tuples $(a,b) \in \textsc{Good}_i$ with $(\sqrt{\theta})^{a+1} P_{\text{start}} < 4 \log n / N$. The above lemma shows that $\sum_{j \in I_i^{(a,b)}} p_j \geq 100 \log n$. Thus, if we mark all the tuples (a,b) with $\boldsymbol{X}_i^{(a,b)} \geq 66 \log n$ as "probably good for i", then with a probability $\geq 1 - 1/n^2$, all the tuples (a,b) in $\textsc{Good}_i$ are marked as "probably good for i" and any tuple $(a,b) \in \textsc{Bad}_i$ marked as "probably good for i" satisfies

$$\boldsymbol{X}_i^{(a,b)} \leq 2.5 \sum_{j \in I_i^{(a,b)}} p_j.$$

We now consider the following algorithm for computing $\hat{x}_i$ using which we then compute $\mathbf{Est}_i$:

1. Let $S_i = \{j \mid i \in \mathbf{SC}_j\}$ be the set of all the servers that sample the coordinate i.
2. Let $\hat{x}_i \leftarrow 0$ denote our initial estimate for x_i.
3. Let $\boldsymbol{L}_i = \{j \in S \mid \frac{\boldsymbol{e}_i^{-1} f(x_i(j))}{\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))} > \frac{4s}{c_f[s] \log^2 n}\}$. This corresponds to the servers in $\textsc{Large}_i$ that have sampled i. With high probability, $\boldsymbol{L}_i = \textsc{Large}_i$. Note that we know the value $x_i(j)$ for all $j \in \boldsymbol{L}_i$.

4. Update $\hat{x}_i \leftarrow \hat{x}_i + \sum_{j \in \boldsymbol{L}_i} x_i(j)$.
5. Update $\boldsymbol{S}_i \leftarrow \boldsymbol{S}_i \setminus \boldsymbol{L}_i$.
6. Let $\mathbf{Sm}_i = \{j \in \boldsymbol{S}_i \mid \frac{\boldsymbol{e}_i^{-1} f(x_i(j))}{\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))} < \frac{\varepsilon_1}{c_f[s] \cdot (C \log^2 n)}\}$. Corresponds to the servers in $\textsc{Small}_i$ that have sampled i and therefore the contribution from these servers can be ignored.
7. Update $\boldsymbol{S}_i \leftarrow \boldsymbol{S}_i \setminus \mathbf{Sm}_i$.
8. For each $a = 0, 1, \ldots, A-1$ and $b = 0, 1, \ldots, B-1$, set $\boldsymbol{S}_i^{(a,b)} \leftarrow \{j \in \boldsymbol{S}_i \mid j \in I_i^{(a,b)}\}$. Note that since for all $j \in \boldsymbol{S}_i$, we know the value of $x_i(j)$ and therefore we can compute the set $\boldsymbol{S}_i^{(a,b)}$.
9. For all tuples (a, b) with $(\sqrt{\theta})^{a+1} P_{\text{start}} > 4 \log n / N$, with high probability $\boldsymbol{S}_i^{(a,b)} = I_i^{(a,b)}$ and we update $\hat{x}_i \leftarrow \hat{x}_i + \sum_{j \in \boldsymbol{S}_i^{(a,b)}} x_i(j)$.
10. For all other tuples (a, b), if $|\boldsymbol{S}_i^{(a,b)}| \geq 66 \log n$, we mark (a, b) as "probably good for i" and ignore the rest of the tuples.
11. For all tuples (a, b) marked as "probably good for i", we update

$$\hat{x}_i \leftarrow \hat{x}_i + \frac{(2/5)|\boldsymbol{S}_i^{(a,b)}|}{1 - (1 - (\sqrt{\theta})^{a+1} P_{\text{start}})^N} f^{-1}(\boldsymbol{e}_i (\sqrt{\theta})^{a+b} P_{\text{start}} F_{\text{start}}).$$

12. Compute $\mathbf{Est}_i \coloneqq \boldsymbol{e}_i^{-1} f(\hat{x}_i)$.

Using all the lemmas we have gone through till now, we will argue that for all $i \in \mathbf{SC}$, $\hat{x}_i \leq x_i$ with a high probability, which proves that $\mathbf{Est}_i$ is upper bounded by $\boldsymbol{e}_i^{-1} f(x_i)$. We then prove $\mathbf{Est}_{i^*}$ is large.

Lemma 13.4.12. *With probability $\geq 1 - 1/n$, for all $i \in [n]$, $\hat{x}_i \leq x_i$ which implies $\mathbf{Est}_i \leq \boldsymbol{e}_i^{-1} f(x_i)$.*

Proof. Consider a fixed i. We will argue about contributions from different types of servers to $\hat{x}_i$ separately. By Lemma 13.4.5, we have $\boldsymbol{L}_i = \textsc{Large}_i$ with probability $\geq 1 - 1/n^3$ and the contribution $\sum_{j \in \textsc{Large}_i} x_i(j)$ to x_i is estimated correctly. We deterministically exclude all the servers in $\textsc{Small}_i$ and therefore we do not overestimate the contribution of $\sum_{j \in \textsc{Small}_i} x_i(j)$ to x_i.

Now consider the tuples (a, b) for $a = 0, 1, \ldots, A-1$ and $b = 0, 1, \ldots, B-1$. If a is such that $(\sqrt{\theta})^{a+1} P_{\text{start}} \geq 4 \log n / N$, we again have $\boldsymbol{S}_i^{(a,b)} = I_i^{(a,b)}$ with a large probability similar to the analysis of Lemma 13.4.5 and therefore we estimate the contribution of such servers to x_i exactly.

If $(\sqrt{\theta})^{a+1} P_{\text{start}} < 4 \log n / N$ and $(a, b) \in \textsc{Good}_i$, then Lemma 13.4.11 shows that $\sum_{j \in I_i^{(a,b)}} p_j \geq 100 \log n$. We then obtain using Lemma 13.4.10 that $66 \log n \leq |\boldsymbol{S}_i^{(a,b)}| \leq (4/3) \sum_{j \in I_i^{(a,b)}} p_j$ with a large probability. Using Lemma 13.4.9, we get that

$$\frac{|\boldsymbol{S}_i^{(a,b)}|}{(4/3)(1 - (1 - (\sqrt{\theta})^{a+1} P_{\text{start}})^N)} \leq |I_i^{(a,b)}|.$$

Using Lemma 13.4.8, we then obtain that

$$\frac{|S_i^{(a,b)}|}{(4/3)(1-(1-(\sqrt{\theta})^{a+1}P_{\text{start}})^N)} f^{-1}(\boldsymbol{e}_i(\sqrt{\theta})^{a+b}P_{\text{start}}F_{\text{start}}) \\ \leq |I_i^{(a,b)}| f^{-1}(\boldsymbol{e}_i(\sqrt{\theta})^{a+b}P_{\text{start}}F_{\text{start}}) \leq \sum_{j \in I_i^{(a,b)}} x_i(j).$$

Hence, the contribution of tuples $(a,b) \in \textsc{Good}_i$ is not overestimated. If $(a,b) \in \textsc{Bad}_i$ and (a,b) is marked "probably good for i", then using Lemma 13.4.10, we get

$$|S_i^{(a,b)}| \leq \frac{5}{2} \sum_{j \in I_i^{(a,b)}} p_j$$

and using the same series of steps as above, we get

$$\frac{|S_i^{(a,b)}|}{(5/2)(1-(1-(\sqrt{\theta})^{a+1}P_{\text{start}})^N)} f^{-1}(\boldsymbol{e}_i(\sqrt{\theta})^{a+b}P_{\text{start}}F_{\text{start}}) \\ \leq |I_i^{(a,b)}| f^{-1}(\boldsymbol{e}_i(\sqrt{\theta})^{a+b}P_{\text{start}}F_{\text{start}}) \leq \sum_{j \in I_i^{(a,b)}} x_i(j)$$

which again shows that the contribution of tuples (a,b) in $\textsc{Bad}_i$ but are marked "probably good for i" is also not overestimated. Overall we get that with a probability $\geq 1 - O(1/n^2)$, $\hat{x}_i \leq x_i$. Using a union bound we have the proof. ◻

We now show that $\textbf{Est}_{i^*}$ is large with a large probability.

Lemma 13.4.13. *With probability* $\geq 1 - 1/n^2$,

$$\textbf{\textit{Est}}_{i^*} \geq \frac{(1-\varepsilon_2)^2}{\theta''} \boldsymbol{e}_{i^*}^{-1} f(x_{i^*}).$$

Proof. By Lemma 13.4.5, $\boldsymbol{L}_{i^*} = \textsc{Large}_{i^*}$ and therefore the contribution to x_{i^*} from the servers in $\textsc{Large}_{i^*}$ is captured by $\hat{x}_{i^*}$. We argued that contribution to $\hat{x}_{i^*}$ from servers in $\textsc{Small}_{i^*} \cup \textsc{Ignore}_{i^*}$ can be ignored.

Now consider the tuples (a,b) for $a = 0, 1, \ldots, A-1$ and $b = 0, 1, \ldots, B-1$. If $(\sqrt{\theta})^{a+1}P_{\text{start}} \geq (4\log n)/N$, then $\boldsymbol{S}_{i^*}^{(a,b)} = I_{i^*}^{(a,b)}$ with probability $\geq 1 - 1/n^3$ and therefore the contribution from the servers in $I_{i^*}^{(a,b)}$ is captured exactly by $\hat{x}_{i^*}$. If $(a,b) \in \textsc{Bad}_{i^*}$, then we argued in Lemma 13.4.7 that we need not capture the contribution from those servers. So any contribution from servers in $\textsc{Bad}_{i^*}$ marked as "probably good for i^*" will only help in increasing $\hat{x}_{i^*}$.

Now consider $(a, b) \in \textsc{Good}_{i^*}$ with $(\sqrt{\theta})^{a+1}P_{\text{start}} < 4\log n/N$. By Lemma 13.4.11, with a large probability $|S_{i^*}^{(a,b)}| \ge \frac{2}{3}\sum_{j \in I_{i^*}^{(a,b)}} p_j$. By Lemma 13.4.9, we get

$$|S_{i^*}^{(a,b)}| \ge \frac{2}{3}|I_{i^*}^{(a,b)}|\frac{1-(1-(\sqrt{\theta})^{a+1}P_{\text{start}})^N}{\sqrt{\theta}}.$$

Now, using Lemma 13.4.8, we get

$$|S_{i^*}^{(a,b)}| \cdot f^{-1}(\boldsymbol{e}_{i^*}(\sqrt{\theta})^{a+b}P_{\text{start}}F_{\text{start}}) \ge \frac{2}{3\cdot\theta'} \cdot \frac{1-(1-(\sqrt{\theta})^{a+1}P_{\text{start}})^N}{\sqrt{\theta}} \sum_{j \in I_{i^*}^{(a,b)}} x_{i^*}(j)$$

which then implies

$$\frac{(2/5)|S_{i^*}^{(a,b)}| \cdot f^{-1}(\boldsymbol{e}_{i^*}(\sqrt{\theta})^{a+b}P_{\text{start}}F_{\text{start}})}{1-(1-(\sqrt{\theta})^{a+1}P_{\text{start}})^N} \ge \frac{4}{15\cdot\theta\cdot\sqrt{\theta}} \sum_{j \in I_{i^*}^{(a,b)}} x_{i^*}(j).$$

Thus, overall, with high probability, we have

$$\hat{x}_{i^*} \ge \sum_{j \in \textsc{Large}_{i^*}} x_{i^*}(j) + \frac{4}{15\cdot\theta\cdot\sqrt{\theta}} \sum_{(a,b)\in\textsc{Good}_{i^*}} \sum_{j \in I_{i^*}^{(a,b)}} x_{i^*}(j).$$

We therefore have

$$\mathbf{Est}_{i^*} \coloneqq \boldsymbol{e}_{i^*}^{-1} f(\hat{x}_{i^*}) \ge \frac{(1-\varepsilon_2)^3}{\theta''}\boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$$

assuming for all x, $f(x/(4\cdot\theta\cdot\sqrt{\theta})) \ge f(x)/\theta''$. □

Theorem 13.4.14. *Assume we are given a super-additive nonnegative function f that satisfies the "approximate invertibility" property with parameters $\theta, \theta', \theta'', \varepsilon_1$ and ε_2. Let $s \ge 1$ be the number of servers. Define $A = O(\log(s/\varepsilon_1)/\log\theta)$ and $B = O(\log(s^2/\varepsilon_1(1-\varepsilon_2))/\log\theta)$. Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ be independent standard exponential random variables shared across all the servers. Then there is a 2-round protocol in the coordinator model which uses a total communication of*

$$O\left(c_f[s] \cdot \frac{c_f[A\cdot B]\cdot(A\cdot B)\cdot\sqrt{\theta}\log^4 n}{\varepsilon_1(1-\varepsilon_2)^2} + s \cdot \frac{\log^2 n \cdot \theta''}{(1-\varepsilon_2)^3}\right)$$

words of communication and with probability $\ge 1 - 1/\text{poly}(n)$ computes $\max_i \boldsymbol{e}_i^{-1} f(x_i)$.

Proof. First we condition on the event that $\sum_i \boldsymbol{e}_i^{-1} f(x_i) \le (C\log^2 n)\cdot\max_i \boldsymbol{e}_i^{-1} f(x_i)$ which holds with probability $\ge 1 - 1/\text{poly}(n)$. Note that the randomness used in sampling is independent of the exponential random variables. Define $i^* \coloneqq \arg\max_i \boldsymbol{e}_i^{-1} f(x_i)$.

Let $N = O((c_f[s]/s) \cdot c_f[A \cdot B] \cdot (A \cdot B) \cdot \sqrt{\theta} \log^4 n/(\varepsilon_1(1-\varepsilon_2)^2))$. Let each server j sample N coordinates independently from its local distribution:

$$\mathbf{Pr}[\boldsymbol{i} = i] = \frac{\boldsymbol{e}_i^{-1} f(x_i(j))}{\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))}.$$

Let SC_j be the set of coordinates sampled by the server j. Now, each server j sends the set SC_j along with the values $x_i(j)$ for $i \in \mathsf{SC}_j$ to the coordinator. Additionally, each server also sends the value $\sum_i \boldsymbol{e}_i^{-1} f(x_i(j))$ to the coordinator. Note that this requires a total communication of $O(N \cdot s)$ words of communication.

Now, the coordinator computes $\mathsf{SC} = \bigcup_j \mathsf{SC}_j$. By Lemma 13.4.3, we have $i^* \in \mathsf{SC}$ with probability $\geq 1 - 1/\text{poly}(n)$. Now the coordinator computes $\hat{x}_i$ and a value $\mathbf{Est}_i$ for each $i \in \mathsf{SC}$ using the algorithm described above. Lemma 13.4.12 shows that with a probability $\geq 1 - 1/\text{poly}(n)$, for all $i \in \mathsf{SC}$, we have $\mathbf{Est}_i \leq \boldsymbol{e}_i^{-1} f(x_i)$ and Lemma 13.4.13 shows that with probability $\geq 1 - 1/\text{poly}(n)$, we have $\mathbf{Est}_{i^*} \geq \frac{(1-\varepsilon_2)^3}{\theta''} \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$. Using a union bound, all these events hold with probability $\geq 1 - 1/\text{poly}(n)$. Condition on all these events.

Let PL be the set of coordinates $i \in \mathsf{SC}$ with the $O(C \log^2 n \cdot \theta''/(1-\varepsilon_2)^3)$ largest values of $\mathbf{Est}_i$. We have $i^* \in \mathsf{PL}$ since we conditioned on all the above events. Now the coordinator queries sends the set PL to *each* server j and asks for the values $x_i(j)$ for $i \in \mathsf{PL}$. Then the servers all send the requested information in the second round of communication. Note that the total communication required is $O(s \cdot |\mathsf{PL}|)$ words. Since i^* in PL, we obtain that

$$\max_{i \in \mathsf{PL}} \boldsymbol{e}_i^{-1} f(\sum_{j \in [s]} x_i(j)) = \boldsymbol{e}_{i^*}^{-1} f(x_{i^*})$$

and the coordinator can compute this value after receiving the required information from the servers in the second round of communication. This proves the theorem. □

We can run the protocol in the above theorem concurrently using $O(1/\varepsilon^2)$ independent copies of the exponential random variables and then obtain a $1 \pm \varepsilon$ approximation for $\sum_i f(x_i)$ with a probability $\geq 99/100$. We note that the overall protocol requires two rounds and a total communication of $O_{\theta,\theta',\theta''}\left(\frac{c_f[s]}{\varepsilon^2} \text{polylog}(n)\right)$ words of communication.

Theorem 13.4.15. *Let f be a non-negative, increasing, super-additive function that satisfies the "approximate invertibility" properties with the parameters $\theta, \theta', \theta''$. Let there be s servers and each of the servers holds a non-negative vector $x(1), \ldots, x(s) \in \mathbb{R}^n$ respectively. Define $A = O(\log(s \cdot \theta'')/\log\theta)$ and $B = O(\log(s^2 \cdot (\theta'')^2)/\log\theta)$. Given $\varepsilon < 1/n^c$ for a small constant c, there is a two round protocol that uses* *a total of*

$$O\left(\frac{c_f[s]}{\varepsilon^2}\cdot c_f[A\cdot B]\cdot(A\cdot B)\cdot\sqrt{\theta}\cdot(\theta'')^3\cdot\log^4 n+\frac{s}{\varepsilon^2}\cdot\log^2 n\cdot(\theta'')^4\right)$$

words of communication and with probability $\geq 9/10$ *computes a* $1\pm\varepsilon$ *for the quantity* $\sum_i f(x_i)$.

Proof. For $k\in[O(1/\varepsilon^2)]$ and $i\in[n]$, let $\boldsymbol{e}_i^{(k)}$ be an independent standard exponential random variable. Let $i^*(k)\coloneqq\arg\max_{i\in[n]}(\boldsymbol{e}_i^{(k)})^{-1}f(x_i)$. By a union bound, the following hold simultaneously with probability $\geq 1-1/\text{poly}(n)$:

$$\text{for all } k,\ \sum_i(\boldsymbol{e}_i^{(k)})^{-1}f(x_i)\leq(C\log^2 n)\cdot(\boldsymbol{e}_{i^*(k)}^{(k)})^{-1}f(x_{i^*(k)})$$

and

$$\ln(2)\cdot\text{median}_k\,(\boldsymbol{e}_{i^*(k)}^{(k)})^{-1}f(x_{i^*(k)})=(1\pm\varepsilon)\sum_{i\in[n]}f(x_i).$$

We condition on these events. Now concurrently for each k, we use the exponential random variables $\boldsymbol{e}_1^{(k)},\ldots,\boldsymbol{e}_n^{(k)}$ and run the protocol in Theorem 13.4.14 to obtain the value of $(\boldsymbol{e}_{i^*(k)}^{(k)})^{-1}f(x_{i^*(k)})$ with probability $\geq 1-1/\text{poly}(n)$. We union bound over the success of the protocol for all k and obtain that with probability $\geq 9/10$, we can compute the exact value of

$$\ln(2)\cdot\text{median}_k\,(\boldsymbol{e}_{i^*(k)}^{(k)})^{-1}f(x_{i^*(k)})$$

which then gives us a $1\pm\varepsilon$ approximation of $\sum_{i\in[n]}f(x_i)$. The communication bounds directly follow from Theorem 13.4.14. □

We obtain the following corollary for estimating F_k moments.

Corollary 13.4.16. *Let* $k>2$ *be arbitrary. In the coordinator model with* s *servers that each hold a non-negative vector* $x(j)\in\mathbb{R}^n$*, there is a randomized two round protocol that uses a total of* $\widetilde{O}_k(s^{k-1}\,\text{polylog}(n)/\varepsilon^2)$ *bits of communication and approximate* $\sum_i(\sum_{j\in[s]}x_i(j))^k$ *up to a* $1\pm\varepsilon$ *factor with probability* $\geq 9/10$.

Proof. For the function $f(x)=x^k$, we have $c_f[s]=s^{k-1}$ by a simple application of the Holder's inequality. We additionally note that x^k is "approximately invertible" with parameters $\theta=2$, $\theta'=2^{1/k}$, and $\theta''=2\cdot 8^{k/2}$. Therefore ε_1 and $1-\varepsilon_2$ can be taken as $1/(2\cdot 8^{k/2})$. We now have $A,B=O(k+\log s)$ so that $c_f[A\cdot B]\cdot(A\cdot B)=(k+\log s)^k$. From the above theorem, we therefore obtain that there is a two round protocol that computes a $1\pm\varepsilon$ approximation of $\sum_{i\in[n]}(\sum_{j\in[s]}x_i(j))^k$ with probability $\geq 9/10$ and uses a total communication of

$$O\left(\frac{s^{k-1}}{\varepsilon^2}\frac{(k+\log s)^k}{8^{3k/2}}\text{polylog}(n)\right)$$

words of communication. □

13.4.3 Higher-Order Correlations

Kannan, Vempala, and Woodruff [KVW14] also study the problem of approximating higher order correlations and list a few applications of the problem in their paper. In this problem, there are s servers and the j-th server holds a set of n-dimensional vectors W_j. Given a parameter k, and functions $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}, g : \mathbb{R}^k_{\geq 0} \to \mathbb{R}_{\geq 0}$, the coordinator wants to approximate

$$M(f, g, W_1, \ldots, W_s) \coloneqq \sum_{i_1, i_2, \ldots, i_k \text{ distinct}} f\left(\sum_j \sum_{v \in W_j} g(v_{i_1}, v_{i_2}, \ldots, v_{i_k})\right)$$

As they mention, for each server j, we can create a vector $w(j)$ with $r = \binom{n}{k}k!$ components (one for each tuple $(i_1, \ldots, i_k)$ with distinct values of $i_1, i_2, \ldots, i_k \in [n]$) defined as

$$[w(j)]_{(i_1,i_2,\ldots,i_k)} \coloneqq \sum_{v \in W_j} g(v_{i_1}, v_{i_2}, \ldots, v_{i_k}).$$

Now running the function sum approximation protocol on the vectors $w(1), \ldots, w(s)$ with the function f, we can compute a $1 \pm \varepsilon$ approximation for $M(f, g)$ using a total of

$$O_{\theta,\theta',\theta''}\left(\frac{c_f[s]}{\varepsilon^2} \cdot \text{polylog}(r)\right) = O_{\theta,\theta',\theta''}\left(\frac{c_f[s]}{\varepsilon^2} \cdot \text{poly}(k, \log n)\right)$$

words of communication. The main issue in implementing this algorithm is that all the servers have to realize the $r = \binom{n}{k}k!$ dimensional vectors $w(j)$ which end up occupying $O(n^k)$ space which is prohibitive when n is large. Using a simple trick, we can show that to execute the protocol in Theorem 13.4.15 can be implemented without using $O(n^k)$ space.

We first solve for the issue of sharing $O(n^k)$ exponential random variables across all the servers. In Appendix E.3, we show that the exponential random variables used in the protocol need not be independent but can be generated using Nisan's Pseudorandom Generator (PRG) [Nis92]. The seed for Nisan's PRG needs to be only of length $O(k^2 \log^2(n/\varepsilon))$ and hence the shared randomness across the servers is only of this size.

The main reason we need the vectors $w(j)$ is so that the server j can sample independent copies of the random variable $(\boldsymbol{i}_1, \ldots, \boldsymbol{i}_k)$ with probability distribution

$$\mathbf{Pr}[(\boldsymbol{i}_1, \ldots, \boldsymbol{i}_k) = (i_1, \ldots, i_k)] = \frac{(\boldsymbol{e}_{(i_1,\ldots,i_k)})^{-1}[w(j)]_{(i_1,\ldots,i_k)}}{\sum_{i'_1,\ldots,i'_k}(\boldsymbol{e}_{(i'_1,\ldots,i'_k)})^{-1}[w(j)]_{(i'_1,\ldots,i'_k)}}$$

where $\boldsymbol{e}_{(i_1,\ldots,i_k)}$ is an independent standard exponential random variable. But we note that to sample from this distribution, the protocol does not need $O(n^k)$ space. Consider the lexicographic ordering of the tuples $(i_1, \ldots, i_k)$ with all $i_1, \ldots, i_k$ distinct. The algorithm goes over the tuples in the lexicographic orders, computes the value of $(\boldsymbol{e}_{(i_1,\ldots,i_k)})^{-1}[w(j)]_{(i_1,\ldots,i_k)}$ by generating the random variable $\boldsymbol{e}_{(i_1,\ldots,i_k)}$ using Nisan's PRG and then uses a *reservoir sampling* algorithm to sample a tuple form the above defined distribution. This entire process can be accomplished using a constant amount of space and hence implementing the function sum approximation protocol on the vectors $w(1), \ldots, w(s)$ can be accomplished without using $\Theta(n^k)$ space at each of the servers. Hence we have the following theorem:

Theorem 13.4.17. *Let there be s servers each holding an arbitrary set of n-dimensional non-negative vectors $W_1, \ldots, W_s \subseteq$ respectively. Given a function $f : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ that satisfies the "approximate invertibility" property with parameters $\theta, \theta', \theta'' > 1$ and a function $g : \mathbb{R}^k_{\ge 0} \to \mathbb{R}^k_{\ge 0}$, there is a randomized two round protocol that approximates $M(f, g, W_1, \ldots, W_s)$ up to a $1 \pm \varepsilon$ factor with probability $\ge 9/10$. The protocol uses a total of*

$$O\left(\frac{c_f[s]}{\varepsilon^2} \cdot c_f[A \cdot B] \cdot (A \cdot B) \cdot \sqrt{\theta} \cdot (\theta'')^3 \cdot k^4 \log^4 n + \frac{s}{\varepsilon^2} \cdot k^2 \log^2 n \cdot (\theta'')^4\right)$$

words of communication, where $A = O(\log(s \cdot \theta'')/\log\theta)$ and $B = O(\log(s^2 \cdot (\theta'')^2)/\log\theta)$

13.5 Lower Bounds

13.5.1 Lower Bound for Sum Approximation

For F_k approximation problem, [WZ12] show an $\Omega(s^{k-1}/\varepsilon^2)$ lower bound on the total communication of any protocol that $1 + \varepsilon$ approximates the F_k value of a vector that is distributed among s servers. They show the lower bound by reducing from a communication problem called the k-BTX. Their proof can be adapted in a straightforward way to obtain the following result for general function approximation for some class of functions f.

Theorem 13.5.1. *Let f be a non-negative, super-additive function and $c_f[s]$ be the parameter such that for all $y_1, \ldots, y_s \ge 0$, then*

$$f(y_1 + \cdots + y_s) \le \frac{c_f[s]}{s}(\sqrt{f(y_1)} + \cdots + \sqrt{f(y_s)})^2.$$

Assume that there exists y^* such that the above inequality is tight when $y_1 = \cdots = y_s = y^*$ i.e., $f(sy^*) = s \cdot c_f[s] \cdot f(y^*)$ and let $\beta = f(sy^*)/(2 \cdot f(sy^*/2)) \geq 1$. If $s \geq \Omega(\beta)$, then any protocol that approximates $\sum_i f(x_i)$ in the coordinator model up to a $1 \pm (1/72\beta - 1/72\beta^2)\varepsilon$ factor must use a total communication of $\Omega(c_f[s]/\varepsilon^2)$ bits.

While the requirements in the above theorem may seem circular, as in, $\beta = f(sy^*)/2f(sy^*/2)$ and $s \geq \Omega(\beta)$, note that β is upper bounded by $\max_x f(x)/(2f(x/2))$ which is independent of the number of servers s.

Proof. We prove the communication lower bound by showing that any protocol which can approximate $\sum_i f(x_i)$ where f is a function that satisfies the properties in the theorem statement can be used to construct a protocol for solving the so-called s-BTX (Block-Threshold-XOR) problem on a specific hard input distribution ν.

To define the s-BTX communication problem and a hard distribution ν, we first define the s-XOR problem and a hard distribution ψ_n for this problem. There are s sites $S_1, \ldots, S_s$. Each site S_j holds a block $b(j) = (b_1(j), \ldots, b_n(j))$ of n bits. The s sites want to compute the following function:

$$s\text{-XOR}(b(1), \ldots, b(s)) = \begin{cases} 1, & \text{if there is an index } i \in [n] \text{ such that} \\ & \quad b_i(j) = 1 \text{ for exactly } s/2 \text{ values of } j, \\ 0, & \text{otherwise.} \end{cases}$$

Woodruff and Zhang [WZ12] define an input distribution φ_n to the s-XOR problem as follows. For each coordinate $i \in [n]$, a variable D_i is chosen uniformly at random from the set $\{1, \ldots, s\}$. Conditioned on the value D_i, all but the D_i-th site sets their input to 0 in the i-th coordinate, whereas the D_i-th site sets its input in the i-th coordinate to 0 or 1 with equal probability. Let φ_1 be this distribution on one coordinate.

Next, a special coordinate M is chosen uniformly at random from $[n]$ and the inputs in the M-th coordinate at all s sites are modified as follows: for the first $s/2$ sites, the inputs in the M-th coordinate are replaced with all 0s with probability $1/2$ and all 1s with probability $1/2$. Similarly, for the last $s/2$ sites, the inputs in the M-th coordinate are replaced with all 0s with probability $1/2$ and all 1s with probability $1/2$. Let ψ_1 denote the input distribution on the special coordinate and ψ_n denote the input distribution that on special coordinate follows ψ_1 and follows φ_1 on the remaining $n-1$ coordinates.

We will now define the s-BTX problem and a hard input distribution ν. Again, there are s sites $S_1, \ldots, S_s$. Each site S_j holds an input consisting of $1/\varepsilon^2$ blocks and each block is an input for that site in a corresponding s-XOR problem. Concretely, each site S_j holds a length n/ε^2 vector $b(j) = (b^1(j), \ldots, b^{1/\varepsilon^2}(j))$ divided into $1/\varepsilon^2$ blocks of n bits each. There are $1/\varepsilon^2$ instances of the s-XOR problem with the ℓ-th instance having the inputs $b^\ell(1), \ldots, b^\ell(s)$. In the s-BTX problem, the sites want to compute the following:

$$s\text{-BTX}(b(1),\ldots,b(s)) = \begin{cases} 1, & \text{if } |\sum_{\ell\in[1/\varepsilon^2]} s\text{-XOR}(b^\ell(1),\ldots,b^\ell(s)) - 1/2\varepsilon^2| \geq 2/\varepsilon \\ 0, & \text{if } |\sum_{\ell\in[1/\varepsilon^2]} s\text{-XOR}(b^\ell(1),\ldots,b^\ell(s)) - 1/2\varepsilon^2| \leq 1/\varepsilon \\ *, & \text{otherwise.} \end{cases}$$

A hard input distribution ν for this problem is defined as follows: The input of the s sites in each block is independently chosen according to the input distribution ψ_n defined above for the s-XOR problem. Let B be the random variable denoting the inputs $(b(1),\ldots,b(s))$ when drawn from input distribution ν and $M = (M^1,\ldots,M^{1/\varepsilon^2})$ denote the random variable where M^ℓ denotes the special coordinate in the ℓ-th block of the inputs and D denotes the special sites for all the coordinates in all $1/\varepsilon^2$ instances of the s-XOR problem. [WZ12] prove the following theorem:

Theorem 13.5.2 ([WZ12, Theorem 7])**.** *Let Π be the transcript of any randomized protocol for the s-BTX problem on input distribution ν with error probability δ for a sufficiently small constant δ. We have $I(B;\Pi \mid M,D) \geq \Omega(n/s\varepsilon^2)$, where information is measured with respect to the input distribution ν.*

The theorem essentially states that the transcript of any protocol that solves the s-BTX problem on the input distribution ν with a large probability must have a large amount of "information" about the input vectors when conditioned on the random variables M, D. Since the randomized communication complexity is always at least the conditional information cost, the above theorem implies that any randomized protocol that solves the s-BTX problem on input distribution ν with error probability δ has a communication complexity of $\Omega(n/s\varepsilon^2)$.

We show a lower bound on the communication complexity of the function sum estimation problem for f in the theorem statement by reducing the s-BTX problem to approximating $\sum_i f(x_i)$ for appropriately chosen vectors $x(1),\ldots,x(s)$ at each of the sites.

Let $n = s \cdot c_f[s]$ so that the communication complexity of a randomized protocol for s-BTX on input distribution ν is $\Omega(c_f[s]/\varepsilon^2)$. Let $(b(1),\ldots,b(s))$ be inputs to the s-BTX problem drawn from the distribution ν. Notice that each $b(j)$ is a binary vector with $s \cdot c_f[s]/\varepsilon^2$ coordinates. Now define $b = b(1) + \cdots + b(s)$.

Since the input $(b(1),\ldots,b(s))$ is drawn from the distribution ν, we note the following about vector b:

1. Each block of $1/\varepsilon^2$ coordinates has exactly one coordinate i in which $b_i = s$ with probability $1/4$, $b_i = s/2$ with probability $1/2$ and $b_i = 0$ with probability $1/4$.
2. In each block, all other coordinates apart from the one singled out above have a value 0 with probability $1/2$ and 1 with probability $1/2$.

Therefore, the vector b when $(b(1),\ldots,b(s))$ is sampled from ν has, in expectation, $\frac{s\cdot c_f-1}{2\varepsilon^2}$ coordinates with value 1, $\frac{1}{2\varepsilon^2}$ coordinates with value $s/2$ and $\frac{1}{4\varepsilon^2}$ coordinates with value s.

For each $j \in [s]$, define $x(j) = y^* \cdot b(j)$ where y^* is as in the theorem statement and let $x = \sum_j x(j) = y^* \cdot b$. From the above properties of the vector b, the vector x has coordinates only with values $0, y^*, sy^*/2, sy^*$ and in expectation it has $\frac{s \cdot c_f[s]-1}{2\varepsilon^2}$ coordinates with value y^*, $1/2\varepsilon^2$ coordinates with value $sy^*/2$ and $1/4\varepsilon^2$ coordinates with value sy^*. So, we write

$$W \coloneqq \sum_i f(x_i) = \left(\frac{s \cdot c_f[s] - 1}{2\varepsilon^2} + Q\right) \cdot f(y^*) + \left(\frac{1}{2\varepsilon^2} + U\right) \cdot f(sy^*/2) + \left(\frac{1}{4\varepsilon^2} + V\right) \cdot f(sy^*)$$

where Q, U, V denote the deviations from the means for each type of coordinate. Note that we have $f(0) = 0$ and hence no contribution from such random variables. Now, the s-BTX problem is exactly to determine if $|U| \geq 2/\varepsilon$ or $|U| \leq 1/\varepsilon$, and we want to show that a protocol to approximate $\sum_i f(x_i)$ can be used to distinguish between the cases.

We now define $x^{\text{left}} = \sum_{j=1}^{s/2} x(j)$ and $x^{\text{right}} = \sum_{j=s/2+1}^{s} x(j)$. Let $W^{\text{left}} \coloneqq \sum_i f(x_i^{\text{left}})$ and $W^{\text{right}} \coloneqq \sum_i f(x_i^{\text{right}})$. We now note that

$$W^{\text{left}} + W^{\text{right}} = \left(\frac{s \cdot c_f[s] - 1}{2\varepsilon^2} + Q\right) \cdot f(y^*) + \left(\frac{1}{2\varepsilon^2} + U\right) \cdot f(sy^*/2) + \left(\frac{1}{4\varepsilon^2} + V\right) \cdot 2 \cdot f(sy^*/2).$$

Note that for the function f, we have $f(sy^*) = \beta \cdot 2 \cdot f(sy^*/2)$ for some $\beta > 1$. Hence,

$$\beta(W^{\text{left}} + W^{\text{right}}) - W = (\beta - 1)\left(\left(\frac{s \cdot c_f[s] - 1}{2\varepsilon^2} + Q\right) \cdot f(y^*) + \left(\frac{1}{2\varepsilon^2} + U\right) \cdot f(sy^*/2)\right).$$

Let $\mathscr{P}$ be a protocol that can approximate $\sum_i f(x_i)$, up to a $1 \pm \alpha\varepsilon$ factor, when the vector x is distributed across s servers. Let $\widetilde{W}$, $\widetilde{W}^{\text{left}}$ and $\widetilde{W}^{\text{right}}$ be the $1 \pm \alpha\varepsilon$ approximations for W, W^{left} and W^{right} computed by running the protocol $\mathscr{P}$ on three different instances of the function sum approximation problem. We first note that for the vector x constructed using the inputs $(b(1), \ldots, b(s))$, we have $\sum_i f(x_i) \leq \frac{s \cdot c_f[s]}{\varepsilon^2} f(y^*) + \frac{1}{\varepsilon^2} f(sy^*) \leq \frac{2 \cdot f(sy^*)}{\varepsilon^2}$ with probability 1 where we used the fact that $s \cdot c_f[s] \cdot f(y^*) = f(sy^*)$. Hence,

$$\widetilde{W} = W \pm \frac{\alpha \cdot 2 \cdot f(sy^*)}{\varepsilon}, \quad \widetilde{W}^{\text{left}} = W^{\text{left}} \pm \frac{\alpha \cdot 2 \cdot f(sy^*)}{\varepsilon}, \quad \text{and} \quad \widetilde{W}^{\text{right}} = W^{\text{right}} \pm \frac{\alpha \cdot 2 \cdot f(sy^*)}{\varepsilon}$$

which then implies

$$\begin{aligned}
&\beta(\widetilde{W}^{\text{left}} + \widetilde{W}^{\text{right}}) - \widetilde{W} \\
&= \beta(W^{\text{left}} + W^{\text{right}}) - W \pm \frac{6\alpha\beta}{\varepsilon} f(sy^*) \\
&= (\beta - 1)\left(\left(\frac{s \cdot c_f[s] - 1}{2\varepsilon^2} + Q\right) \cdot f(y^*) + \left(\frac{1}{2\varepsilon^2} + U\right) \cdot f(sy^*/2)\right) \pm \frac{6\alpha\beta}{\varepsilon} f(sy^*).
\end{aligned}$$

Now, we note that with a large constant probability over the distribution ν, the random variable Q satisfies

$$|Q| \le \frac{C\sqrt{s \cdot c_f[s]}}{\varepsilon}$$

for a large enough constant C by a simple application of a Chernoff bound. Hence, with a union bound on the above event and the correctness of the protocol on inputs x, x^{left} and x^{right}, we get

$$\begin{aligned}\beta(\widetilde{W}^{\text{left}} + \widetilde{W}^{\text{right}}) - \widetilde{W} &= (\beta - 1) \cdot \left(\frac{1}{2\varepsilon^2} + U\right) \cdot f(sy^*/2) + (\beta - 1) \cdot \left(\frac{s \cdot c_f[s] - 1}{2\varepsilon^2}\right) \cdot f(y^*) \\ &\pm (\beta - 1)\frac{C\sqrt{s \cdot c_f[s]}}{\varepsilon} f(y^*) \pm \frac{6\alpha\beta}{\varepsilon} f(sy^*).\end{aligned}$$

Dividing the expression by $(\beta - 1)$, we get

$$\begin{aligned}\frac{\beta(\widetilde{W}^{\text{left}} + \widetilde{W}^{\text{right}}) - \widetilde{W}}{\beta - 1} &= \left(\frac{1}{2\varepsilon^2} + U\right) \cdot f(sy^*/2) + \left(\frac{s \cdot c_f[s] - 1}{2\varepsilon^2}\right) f(y^*) \\ &\pm \frac{C\sqrt{s \cdot c_f[s]}}{\varepsilon} f(y^*) \pm \frac{6\alpha\beta}{\varepsilon(\beta - 1)} f(sy^*)\end{aligned}$$

We now use $f(sy^*) = s \cdot c_f[s] \cdot f(y^*)$, $f(sy^*/2) = s \cdot c_f[s] \cdot f(y^*)/2\beta$ to obtain that

$$\frac{\beta(\widetilde{W}^{\text{left}} + \widetilde{W}^{\text{right}}) - \widetilde{W}}{\beta - 1} = \frac{s \cdot c_f[s] \cdot f(y^*)}{2\beta} \cdot \left(\frac{1}{2\varepsilon^2} + U + \frac{(s \cdot c_f[s] - 1) \cdot 2\beta}{2\varepsilon^2 \cdot s \cdot c_f[s]} \pm \frac{2C \cdot \beta}{\varepsilon\sqrt{s \cdot c_f[s]}} \pm \frac{12\alpha\beta^2}{\varepsilon(\beta - 1)}\right).$$

If $s \ge C' \cdot \beta$, then $\sqrt{s \cdot c_f[s]} \ge s \ge C' \cdot \beta$ as well. If $C' \ge 8C$, and $\alpha \le (\beta - 1)/72\beta^2$, then

$$\frac{\beta(\widetilde{W}^{\text{left}} + \widetilde{W}^{\text{right}}) - \widetilde{W}}{\beta - 1} = \frac{s \cdot c_f[s] \cdot f(y^*)}{2\beta} \cdot \left(\frac{1}{2\varepsilon^2} + U + \frac{(s \cdot c_f[s] - 1) \cdot 2\beta}{2\varepsilon^2 \cdot s \cdot c_f[s]} \pm \frac{5}{12\varepsilon}\right).$$

Hence, we can distinguish between the case when $|U| \le 1/\varepsilon$ or $|U| \ge 2/\varepsilon$ using the expression on the LHS of the above equality. As, $n = s \cdot c_f[s]$, the lower bound for the s-BTX problem implies that any randomized protocol that approximates $\sum_i f(x_i)$ in the coordinator model when the vector x is split between s servers, up to a $1 \pm \left(\frac{1}{72\beta} - \frac{1}{72\beta^2}\right)\varepsilon$ factor, with probability $\ge 1 - \delta$ for a small enough constant δ, must use a total communication of $\Omega(c_f[s]/\varepsilon^2)$ bits. □

13.5.2 F_k Estimation Lower Bound for one-round Algorithms

We use the multiplayer set disjointness problem to show that a one round protocol for F_k estimation using shared randomness requires a total of $\widetilde{\Omega}(s^{k-1}/\varepsilon^k)$ bits of communication. In the one-way *blackboard* private-coin communication model, it is known that the t-player *promise* set disjointness problem, with sets drawn from $[n]$, has a communication lower bound of $\Omega(n/t)$. In this problem, each of the t servers receives a subset of $[n]$ with the promise that the sets received by all the servers are either mutually disjoint or that there is exactly one element that is present in all the subsets.

As defined in the introduction, a one-round protocol in the coordinator model can be *implemented* in the standard 1-way blackboard model: In this model, all the servers in a deterministic order *write* the information on a publicly viewable blackboard. The total communication in this model is then the total number of bits written on the blackboard. The one round algorithms in the coordinator model are strictly weaker as each server sends its information to the coordinator without even looking at others bits. The lower bound of $\Omega(n/t)$ in the 1-way blackboard model was shown in [CKS03] and later [Gro09] extended the $\Omega(n/t)$ lower bound to an arbitrary number of rounds.

Theorem 13.5.3. *Given $s \geq 3$ servers each having an n dimensional vector $x(1), \ldots, x(s)$ respectively, any 1-round F_k estimation algorithm, in which the servers send a single message to the coordinator, that approximates $F_k(x)$ up to $1 \pm \varepsilon$ factor with probability $\geq 9/10$ over the randomness in the protocol, must use $\Omega_k(s^{k-1}/\varepsilon^k \log(s/\varepsilon))$ bits of total communication.*

Proof. The lower bounds in [CKS03, Gro09] hold even with shared randomness, but are not stated that way, so one can also argue as follows to handle shared randomness: suppose there is a public coin algorithm in the 1-way blackboard communication model using a total of c bits of communication. Then by Newman's equivalence [New91] of private-coin vs public-coin protocols up to an additive logarithmic increase in the communication, there is a private coin algorithm in the one-way blackboard communication model using a total of $c + O(\log(nt))$ bits. The first player samples one of the strings pre-shared among all the servers and announces the index of the string on the blackboard and then the remaining servers proceed with the computation using this string as the shared random bits. Hence, $c = \Omega(n/t) - O(\log nt)$ and when $t \leq n^{\alpha}$ for a constant $\alpha < 1$, we obtain that $\Omega(n/t)$ bits is a lower bound on the communication complexity of 1-way public coin protocols in the blackboard model that solve the s-player set disjointness problem.

In our model, the F_k estimation algorithm is even weaker than the 1-way public coin protocol in the blackboard model as all the servers send their bits to the coordinator without looking at others bits. Hence, the lower bound of $\Omega(n/t)$ bits can be used to lower bound the communication complexity.

Let $n = s^k/\varepsilon^k$ and $t = s/2$. Consider the instance of a t player set-disjointness problem. We will encode the problem as approximating the F_k moment of an n dimensional vector distributed over s servers.

For $j = 1, \ldots, s/2$, the player j encodes the subset $S_j \subseteq [n]$ they receive as an n dimensional vector $x(j)$ by putting 1 in the coordinates corresponding to the items in the set and 0 otherwise.

Now each of the $s/2$ servers runs a $(1/Cn)$-error protocol in the coordinator model (as in the protocol fails with probability at most $1/Cn$) to approximate $\| \sum_{j=1}^{s} x(j)\|_k^k$ and sends the transcript to the central coordinator. The central coordinator chooses appropriate vectors $x(s/2+1), \ldots, x(s)$ and using the transcripts from the $s/2$ servers finds a $1+\varepsilon$ approximation to $\| \sum_{j=1}^{s} x(j)\|_k^k$ by running the $(1/Cn)$-error protocol for estimating F_k moments.

Let $\| \sum_{j=1}^{s/2} x(j)\|_k^k = T$. Fix an index $i \in [n]$. The central coordinator creates the vectors $x(s/2+1), \ldots, x(s/2+s/2)$ to be all be equal and have a value of $2/\varepsilon$ in coordinate i and remaining positions have value 0. Now consider a NO instance for the set disjointness problem. Then we have $\| \sum_{j=1}^{s} x(j)\|_k^k \le (T-1) + (s/\varepsilon+1)^k$.

Let T' be such that $(1-\varepsilon')T \le T' \le (1+\varepsilon')T$. Then,

$$(1+\varepsilon')\| \sum_{j=1}^{s} x(j)\|_k^k \le \frac{1+\varepsilon'}{1-\varepsilon'}T' - (1+\varepsilon') + (1+\varepsilon')(s/\varepsilon+1)^k.$$

Now consider a YES instance. If all the sets intersect in i, then $\| \sum_{j=1}^{s} x(j)\|_k^k = T - (s/2)^k + (s/\varepsilon + s/2)^k$. Now,

$$(1-\varepsilon')\| \sum_{i=1}^{s} x(j)\|_k^k \ge \frac{1-\varepsilon'}{1+\varepsilon'}T' - (1-\varepsilon')(s/2)^k + (1-\varepsilon')(s/\varepsilon+s/2)^k.$$

If

$$\frac{1-\varepsilon'}{1+\varepsilon'}T' - (1-\varepsilon')(s/2)^k + (1-\varepsilon')(s/\varepsilon+s/2)^k > \frac{1+\varepsilon'}{1-\varepsilon'}T' - (1+\varepsilon') + (1+\varepsilon')(s/\varepsilon+1)^k,$$

we have a test for set disjointness. The above is implied by

$$(1-\varepsilon')(s/\varepsilon+s/2)^k - (1+\varepsilon')(s/\varepsilon+1)^k - (1-\varepsilon')(s/2)^k \ge \frac{4\varepsilon'}{1-(\varepsilon')^2}T'$$

which is further implied by $(1-\varepsilon')(s/\varepsilon+s/2)^k - (1+\varepsilon')(s/\varepsilon+1)^k - (1-\varepsilon')(s/2)^k \ge 8\varepsilon' T$. As $T \le (s/\varepsilon)^k + (s/2)^k$, we obtain that the above is implied by

$$(1-\varepsilon')(1/\varepsilon+1/2)^k - (1+\varepsilon')(1/\varepsilon+1/s)^k - (1-\varepsilon')(1/2^k) \ge 8\varepsilon'(1/\varepsilon^k + 1/2^k).$$

For $s \ge 3$,

$$\left(\frac{1/\varepsilon+1/2}{1/\varepsilon+1/s}\right)^k \ge \left(\frac{1/\varepsilon+1/2}{1/\varepsilon+1/3}\right)^k \ge (1+\varepsilon/8).$$

Hence, setting $\varepsilon' = \varepsilon/C$ for a large enough constant implies that

$$(1-\varepsilon')(1/\varepsilon+1/2)^k - (1+\varepsilon')(1/\varepsilon+1/s)^k \ge \frac{\varepsilon}{16}(1/\varepsilon+1/2)^k$$

For $k \geq 2$, we further get

$$(1-\varepsilon')(1/\varepsilon+1/2)^k - (1+\varepsilon')(1/\varepsilon+1/s)^k - (1-\varepsilon')(1/2^k) \geq \frac{\varepsilon}{32}(1/\varepsilon+1/2)^k.$$

By picking C large enough, we obtain $(\varepsilon/32)(1/\varepsilon+1/2)^k \geq 8\varepsilon'(1/\varepsilon^k+1/2^k)$. Thus, if a $1 \pm \varepsilon'$ approximation of $\| \sum_{j=1}^{s} x(j)\|_k^k$ for any $i \in [n]$ (note that the vectors $x(s/2+1), \ldots, x(s)$ depend on which i we are using) exceeds $(1+\varepsilon')T'/(1-\varepsilon') - (1+\varepsilon') + (1+\varepsilon')(s/\varepsilon+1)^k$, then we can output YES to the set disjointness instance and otherwise output NO. Note that we needed to union bound over the $n+1$ instances of the problem, i.e., that we compute T' such that $(1-\varepsilon')T \leq T' \leq (1+\varepsilon')T$ and later for each $i \in [n]$, we want a $1 \pm \varepsilon'$ approximation to the appropriately defined $\| \sum_{j=1}^{s} x(j)\|_k^k$ and hence we use a $1/Cn$ error protocol.

Thus, any distributed protocol which outputs a $1+\varepsilon/C$ approximation to the F_k approximation problem with probability $\geq 1 - \varepsilon^k/Cs^k$ must use a total communication of $\Omega_k(s^{k-1}/\varepsilon^k)$ bits. Consequently, an algorithm which succeeds with a probability $\geq 9/10$ must use $\Omega_k(s^{k-1}/\log(s/\varepsilon)\varepsilon^k)$ bits of total communication since the success probability of such an algorithm can be boosted to a failure probability $O(\varepsilon^k/s^k)$ by simultaneous independent copies of the protocol. □

13.6 Conclusions and Open Questions

In this chapter, given a non-negative monotonic function f, we introduce a new parameter $c_f[s]$ and obtain an algorithm for approximating $\sum_i f(x_i)$ in the coordinator model using $c_f[s]/\varepsilon^2$ bits of communication up to polylogarithmic factors. For a restricted class of functions, we show that $\Omega(c_f[s]/\varepsilon^2)$ bits of communication is necessary.

The tightness of our algorithm against the lower bounds suggests that $c_f[s]$ may be the correct parameter to look at compared to the $c_{f,s}$ introduced in the work of Kannan, Vempala and Woodruff [KVW14]. A more careful study is required to fully understand why $c_f[s]$ seems to be capturing the communication complexity of the distributed function sum approximation problem.

Another interesting open question is if we can strengthen the lower bound to showing that $\Omega(c_f[s]/\varepsilon^2)$ bits of communication is necessary for a broader class of functions.

Chapter 14

Linear Algebra in the Personalized CONGEST Model

14.1 Introduction

While a majority of work in the distributed algorithms setting has focussed on the coordinator model, a number of works, such as [CRR14, CLLR17], have looked at more general network topologies. One challenge in a more general network topology and without a coordinator is how to formally define the communication model. One of the most popular distributed frameworks in the past few decades is the CONGEST model [Pel00]. In this model, each server is a node in the network topology and in each round it can send and simultaneously receive a possibly distinct message of bounded size[1] to and from its neighbors, as defined by the edges in the network, which is an unweighted undirected graph. Given the restriction on the size of the messages that can be sent in each round, efficiency of a protocol in this model is in general measured by the number of rounds required by an algorithm.

Efficient CONGEST algorithms have been developed for shortest paths [GL18, HL18], independent sets [Lub86, Gha19], matchings [AKO18, BEPS16], minimum spanning trees [KP98, GKS17], and so on. A related distributed computation model is the on-device public-private model [EEM19] that provides a framework for distributed computation with privacy considerations.

It is not hard to see that one cannot estimate F_k of the sum of vectors as efficiently in the CONGEST model as one can in the coordinator model with s servers, even if one has a two-level rooted tree where each non-leaf node has s children. Indeed, the root and the s nodes in the middle layer can have no input, at which point the problem reduces to the coordinator model with s^2 servers, and for which a stronger $\Omega((s^2)^{k-1})$ lower bound holds [WZ12] and hence the average communication per node in the tree must be $\Omega(s^{2k-4})$ bits, as opposed to an average of $O(s^{k-2})$ bits per server in the coordinator model with s servers. A natural question is which functions can be estimated efficiently with a more general network topology. Inspired by connections between frequency moment algo-

[1]Usually $O(\log n)$ bits where n is the number of nodes in the graph.

rithms and randomized linear algebra (see, e.g., [WZ13]), we study the feasibility of communication efficient algorithms for various linear algebra problems in this setting.

We assume that each server v in the graph holds a matrix $A_v \in \mathbb{R}^{n_v \times d}$. Accordingly, we restrict the size of messages in each round to be $\mathrm{poly}(d, \log n, \log \max_v n_v)$ bits where n is the number of nodes in the graph.

We define a generalization of the CONGEST model called the *personalized CONGEST model.* In this model, given a distance parameter Δ, we would like, after Δ communication rounds, for each node to compute a function of all nodes reachable from it in at most Δ steps, which is referred to as its Δ-hop neighborhood. Note that taking Δ to be the diameter of the graph, we obtain the CONGEST model. Such personalized solutions are desired in several applications such as recommendation systems and online advertisements. For instance, in an application that wants to recommend a restaurant to a user, the data from devices in a different country with no relation to that particular user may not provide useful information and may even introduce some irrelevant bias. Distributed problems with personalization have been studied in a line of work [PHC07, CEK+15, EEM19]; however, to the best of our knowledge none of the prior work studies linear algebraic problems.

Ideally one would like to "lift" a communication protocol for the coordinator model to obtain algorithms for the personalized CONGEST model. However, several challenges arise. The first is that if you have a protocol in the coordinator model which requires more than one round, one may not be able to compute a function of the Δ-hop neighborhood in only Δ rounds. Also, the communication may become too large if a node has to send different messages for each node in say, its 2-hop neighborhood. Another issue is that in applications, one may be most interested in the maximum communication any node has to send, as it may correspond to an individual device, and so cannot be used for collecting a lot of messages and forwarding them. More subtly though, a major issue arises due to multiple distinct paths between two nodes u and v in the same Δ-hop neighborhood. Indeed, if a server is say, interested in a subspace embedding of the union of all the rows held among servers in its Δ-hop neighborhood, we do not want to count the same row twice, but it may be implicitly given a different weight depending on the number of paths it is involved in.

Distributed algorithms for problems in randomized linear algebra, such as regression and low rank approximation, are well studied in the coordinator model [BLS+15, BWZ16, KVW14, FSS20, BKLW14], but they do not work for communication networks with a general topology such as social networks, mobile communication networks, the internet, and other networks that can be described by the CONGEST model. Hence, we ask:

Question For which of the problems in numerical linear algebra can one obtain algorithms in the personalized CONGEST model?

Problem	Per node Communication in each round
ℓ_p subspace embeddings ($p \neq 2$)	$\widetilde{O}_\Delta(d^{\max(p/2+2,3)}\varepsilon^{-2})$ (Theorem 14.3.5)
ℓ_p regression ($p \neq 2$)	$\widetilde{O}_\Delta(d^{\max(p/2+2,3)}\varepsilon^{-2})$ (Section 14.3.5)
ℓ_2 subspace embeddings	$\widetilde{O}_\Delta(d^2\varepsilon^{-2})$ (Theorem 14.3.5)
ℓ_2 regression	$\widetilde{O}_\Delta(d^2\varepsilon^{-2})$ (Section 14.3.5)
Rank-k Frobenius LRA	$\widetilde{O}_\Delta(kd\varepsilon^{-3})$ (Section 14.3.6)

Table 14.1: Per node communication in each of the Δ rounds to solve the problems over data in a Δ neighborhood of each node in the CONGEST model. We assume $\varepsilon < 1/\Delta$.

14.1.1 Our Results

For the above question, in the personalized CONGEST model, we show how to compute Δ-hop subspace embeddings, approximately solve ℓ_p-regression, and approximately solve low rank approximation efficiently. For example, for ℓ_p-subspace embeddings and regression, we achieve $\widetilde{O}(\Delta^2 nd^{p/2+2}/\varepsilon^2)$ words of communication, which we optimize for $p = 2$ to $\widetilde{O}(\Delta^2 nd^2)$ words, where n is the total number of nodes in the graph. Our algorithms are also efficient in that each node sends at most $\widetilde{O}(\Delta \cdot d^{\max(p/2+2,3)})$ words to each of its neighbors in each of the rounds, which we optimize to $\widetilde{O}(\Delta \cdot d^2)$ words for $p = 2$. That is, the maximum communication per server is also small. We remark that in a round, the information sent by a node to all its neighbors is the same.

Finally, our protocols are efficient, in that the total time, up to logarithmic factors is proportional to the number of non-zero entries across the servers, up to additive poly(d/ε) terms. Our results hold more broadly for sensitivity sampling for any optimization problem, which we explain in Section 13.1.2. Our results in the CONGEST model are summarized in Table 14.1.

14.1.2 Our Techniques

In this section, we will describe the construction of ℓ_2-subspace embeddings, though the arguments here are analogous for ℓ_p-subspace embeddings by using the ℓ_p-sensitivities instead of the leverage scores. Recall that given an $n \times d$ matrix A, we say that a matrix M is a $(1/2)$ subspace embedding for A if for all vectors x, $\|Mx\|_2^2 = (1 \pm 1/2)\|Ax\|_2^2$. Subspace embeddings have numerous applications in obtaining fast algorithms for problems such as linear regression, low rank approximation, etc.

Our main technique is to use the *same* uniform random variables across all the servers to *coordinate* the random samples across all the servers in a useful way. For simplicity, assume that there is a node α connected to s neighbors (servers) such that the j-th neighbor holds a matrix $A^{(j)} \in \mathbb{R}^{n_j \times d}$, which does not have any duplicate rows. Further, assume that we want to compute a subspace embedding for the matrix A obtained by the union of the rows of the matrices $A^{(j)}$, i.e., if a row v is present in say both $A^{(1)}$ and $A^{(2)}$, it appears only once in the matrix A.

Given a matrix A, we recall the standard definition of the leverage scores for each of the rows of A, as well as the standard construction to obtain a subspace embedding from the leverage scores. If v is a row of the matrix A, define the leverage score $\tau_A(v)$ of v to be: $\tau_A(v) \coloneqq \max_{x:Ax\neq 0} \frac{|\langle v,x\rangle|^2}{\|Ax\|_2^2}$. For convenience, we define $\tau_A(v) = 0$ if v is not a row of the matrix A. One has that the sum of the leverage scores of all the rows in a matrix A is at most d. Now construct the $n \times n$ random diagonal matrix $\boldsymbol{D}$ as follows: for each $i \in [n]$ independently set $\boldsymbol{D}_{i,i}$ to be $1/\sqrt{p_i}$ with probability $p_i = \min(1, q_i)$ where $q_i \geq C\tau_A(a_i) \log d$ and a_i is the i-th row of A, and set it to 0 otherwise. We note that since $\sum_{i\in[n]} \tau_A(a_i) \leq d$, the random matrix $\boldsymbol{D}$ has at most $O(d \log d)$ nonzero entries with a large probability. One can now show that if C is large enough, then with probability $\geq 99/100$, the matrix $\boldsymbol{D}$ is a $1/2$ subspace embedding for A. We note that the matrix $\boldsymbol{D}A$ has at most $O(d \log d)$ nonzero rows with a large probability. This algorithm is known as *leverage score sampling*.

Going back to our setting, we want to implement leverage score sampling on the matrix A which is formed by the *union* of the rows of the matrices $A^{(1)}, \ldots, A^{(s)}$, i.e., we only count a row once even if it appears on multiple servers. As in the coordinator model in Chapter 13, where we used the same exponential random variables across servers, a key idea we use in the CONGEST model is *correlated randomness*. This time, for each possible row v that could be held by any server, all the servers choose the same threshold $h(v)$ uniformly at random from the interval $[0, 1]$. We treat $h(\cdot)$ as a fully random hash function mapping row v to a uniform random number from the interval $[0, 1]$.

Each server j now computes the ℓ_2 leverage score of each of the rows in its matrix $A^{(j)}$. Note if v is held by two different servers $j \neq j'$, then it could be that $\tau_{A^{(j)}}(v) \neq \tau_{A^{(j')}}(v)$.

Server j sends all its rows v that satisfy $h(v) \leq C\tau_{A^{(j)}}(v) \log d$ to node α. Additionally, assume that the server sends the value $\tau_{A^{(j)}}(v)$ along with the row v. Since $h(v)$ is picked uniformly at random from the interval $[0, 1]$, the probability that a row v is sent to the node α by the j-th server is $\min(1, C\tau_j(v) \log d)$. Hence, each server is implementing leverage score sampling of its own rows and sending all the rows that have been sampled to node α.

Now we note that if v is a row of the matrix $A^{(j)}$, then $\tau_{A^{(j)}}(v) \geq \tau_A(v)$ which directly follows from the definition of leverage scores. For any v that is a row of the matrix A, we have $\tau_A(v) \leq \max_{j\in[s]} \tau_{A^{(j)}}(v)$. Since $h(v)$ is same across all the servers, then the probability that a row v of the matrix A is sent to the node α is exactly, $\min(1, \max_{j\in[s]} C\tau_{A^{(j)}}(v) \log d)$. For all the rows v that are received by node α, it can also compute $\min(1, \max_{j\in[s]} C\tau_{A^{(j)}}(v) \log d)$ since it also receives the values $\tau_{A^{(j)}}(v)$ from all the servers that send the row v. Now using the fact that $\tau_A(v) \leq \max_{j\in[s]} \tau_{A^{(j)}}(v)$ for all rows v of A, the union of rows that are received by the node α correspond to a leverage score sampling of the matrix A. Since the node α can also compute the probability that each row it receives was sampled with, it can appropriately scale the rows and obtain a subspace embedding for the matrix A. In the above procedure, each server j sends at most $O(d \log d)$ rows and therefore the subspace embedding constructed by the node α for matrix A has at most $O(sd \log d)$ rows.

Even though we described a procedure to compute a subspace embedding of the union of neighboring matrices at a single node α, if the nodes send the rows that are under the threshold $h(v)$

to all their neighbors, this procedure can simultaneously compute a subspace embedding at each node for a matrix that corresponds to the union of neighbor matrices of that node. This solves the 1-neighborhood version of the more general Δ-neighborhood problem we introduced.

Now consider how we can compute a subspace embedding for the distance 2 neighborhood matrix. Note that we cannot run the same procedure on 1-neighborhood subspace embeddings to obtain 2-neighborhood subspace embeddings. We again use the monotonicity of leverage scores. Consider the node α and the matrix A as defined before. Let v be a row that it receives from one of its neighbors in the first round. Suppose the node α can compute $\tau_A(v)$, the leverage score of v with respect to the matrix A. Now the node α forwards the row v to its neighbors if $h(v) \le C\tau_A(v)\log d$. Suppose β is neighbor of the node α. Thus, after the second round, the rows received by β then correspond to performing a leverage score sampling of the distance-2 neighborhood matrix for the node β and it can then compute a subspace embedding for that matrix!

Now the main question is how can the node α compute the leverage scores $\tau_A(v)$? By definition of a subspace embedding, we note that if M is a subspace embedding for A, then M can be used to approximate $\tau_A(v)$. Since we already saw that the node α can compute a subspace embedding for the 1-neighborhood matrix, it can also approximate $\tau_A(v)$ for all the rows v that it receives. However, an issue arises where we are using the set of rows that α receives in the first round themselves to approximate their leverage scores and therefore their sampling probabilities in the second round. This leads to correlations, and it is unclear how to analyze leverage score sampling with such correlations. To solve for this issue, we use two independent hash functions $h_1(\cdot)$ and $h_2(\cdot)$. Using the sample of rows received by the node α when the 1-neighborhood procedure from above is run using hash function $h_1(\cdot)$, it computes a subspace embedding for the matrix A and then uses this subspace embedding to approximate the leverage scores of the rows that it receives when the 1-neighborhood procedure run using hash function $h_2(\cdot)$. The node α then uses these approximate leverage scores to decide which of the rows that it received are to be forwarded to its neighbors. This decouples the probability computation and the sampling procedure and the proof of leverage score sampling goes through.

This procedure is similarly extended to compute subspace embeddings for the Δ-neighborhood matrices at each node in the graph. In each round, we use a fresh subspace embedding and use it to compute approximate leverage scores and filter out the rows and then forward them to the neighbors. This way of decorrelating randomness is similar to the *sketch switching* method for adversarial streams in [BJWY22], though we have not seen it used in this context.

This general procedure of collecting data from neighbors, *shrinking* the collected data and transferring the data to all the neighbors is called "graph propagation". Any procedure such as ours above which can handle duplicates can be readily applied in this framework so that each node in the graph can simultaneously learn some statistic/solve a problem over the data in its neighborhood.

14.2 Neighborhood Propagation via Composable Sketches

We define composable sketches and show how using a neighborhood propagation algorithm, composable sketches can be used so that all nodes in a graph with arbitrary topology can simultaneously compute statistics of the data in a distance Δ neighborhood of the node. Typically, the distance parameter Δ is taken to be a small constant but can be as large as the diameter of the underlying graph.

We use $\mathscr{A}$ to denote a dataset. Each item in the dataset is of the form (key, val) where the keys are drawn from an arbitrary set T and the values are d-dimensional vectors. We use the notation $\mathscr{A}$.vals to denote the matrix with rows given by the values in the dataset. We say two datasets $\mathscr{A}$ and $\mathscr{B}$ are *conforming* if for all keys present in both the datasets, the corresponding vals in both the datasets are the same. We use the notation $\mathscr{A} \cup \mathscr{B}$ to denote the union of both the datasets. In the following, we assume that all the relevant datasets are conforming. A composable sketch $\text{sk}(\mathscr{A})$ is a summary of the data items $\mathscr{A}$. The sketch $\text{sk}(\cdot)$ must support the following three operations:

1. $\textsc{Create}(\mathscr{A})$: given data items $\mathscr{A}$, generate a sketch $\text{sk}(\mathscr{A})$.
2. $\textsc{Merge}(\text{sk}(\mathscr{A}_1), \text{sk}(\mathscr{A}_2), \cdots, \text{sk}(\mathscr{A}_k))$: given the sketches $\text{sk}(\mathscr{A}_1), \cdots, \text{sk}(\mathscr{A}_k)$ for sets of data items $\mathscr{A}_1, \cdots, \mathscr{A}_k$ which may have overlaps, generate a composable sketch $\text{sk}(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_k)$ for the union of data items $\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_k$.
3. $\textsc{Solve}(\text{sk}(\mathscr{A}))$: given a sketch $\text{sk}(\mathscr{A})$ of data items $\mathscr{A}$, compute a solution for a pre-specified problem with respect to $\mathscr{A}$.

Note that $\text{sk}(\mathscr{A})$ need not be unique and randomization is allowed during the construction of the sketch and merging. We assume that $\textsc{Create}$, $\textsc{Merge}$ and $\textsc{Solve}$ procedures have access to a shared uniform random bit string.

A core property of the above composable sketch definition is that it handles duplicates. Consider the following problem over a graph $G = (V, E)$. Each vertex of the graph represents a user/server. For a node u, we represent their dataset with $\mathcal{S}_u$, a set of (key, val) pairs. Given a parameter Δ, *each* node in the graph wants to compute statistics or solve an optimization problem over the data of all the nodes within a distance Δ from the node. For example, with $\Delta = 1$, each node u may want to solve a regression problem defined by the data at node u and all the neighbors u.

As composable sketches handle duplicates, the following simple algorithm can be employed to solve the problems over the Δ neighborhood of each node u.

1. Each node u computes $\text{sk}(\mathcal{S}_u)$ and communicates to all its neighbors.
2. Repeat Δ rounds: in round i, each node u computes

$$\text{sk}(\mathcal{S}_u^i) = \text{sk}(\bigcup_{v:\{v,u\}\in E} \mathcal{S}_v^{i-1}) = \textsc{Merge}(\text{sk}(\mathcal{S}_{v_1}^{i-1}), \cdots, \text{sk}(\mathcal{S}_{v_k}^{i-1}))$$

and sends the sketch to all its neighbors. Here we use $\text{sk}(\mathcal{S}_u^0)$ to denote $\text{sk}(\mathcal{S}_u)$.

3. Each node u in the graph outputs a solution over its Δ neighborhood via first computing $\textsc{Merge}(\text{sk}(\mathcal{S}_u^0), \text{sk}(\mathcal{S}_u^1), \text{sk}(\mathcal{S}_u^2), \ldots, \text{sk}(\mathcal{S}_u^\Delta))$ and then using the $\textsc{Solve}(\cdot)$ procedure.

Notice that the capability of handling duplicates is crucial for the above neighborhood propagation algorithm to work. For example, a node v at a distance 2 from u maybe connected to u through two disjoint paths and hence u receives the sketch of u's data from two different sources. So it is necessary for the sketch to be duplicate agnostic to not overweight data of vertices that are connected through many neighbors. Another nice property afforded by composable sketches is that a node sends the same "information" to all its neighbors meaning that a node does not perform different computations determining what information is to be sent to each of its neighbors.

In the following section, we give a composable sketch for computing an ℓ_p subspace embedding and show that it can be used to solve ℓ_p regression problems as well as the low rank approximation problem.

14.3 Composable Sketches for Sensitivity Sampling

We assume $\mathcal{A}_1, \ldots, \mathcal{A}_s$ are conforming datasets. Let $\mathcal{A} := \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_s$. We give a composable sketch construction such that using $\text{sk}(\mathcal{A})$, we can compute an ℓ_p subspace embedding for the matrix $\mathcal{A}.\text{vals}$. Another important objective is to make the size of the sketch $\text{sk}(\mathcal{A})$ as small as possible so that sketches can be efficiently communicated to neighbors in the neighborhood propagation algorithm.

Given a matrix $A \in \mathbb{R}^{n \times d}$, we extend the normal usage and say that a matrix $M \in \mathbb{R}^{m \times d}$ is an ε ℓ_p-subspace embedding for A if for all $x \in \mathbb{R}^d$,

$$\|Mx\|_p^p = (1 \pm \varepsilon)\|Ax\|_p^p.$$

We will now recall the so-called ℓ_p sensitivities, defined in Chapter 2, and how they can be used to compute subspace embeddings.

14.3.1 ℓ_p Sensitivity Sampling

The ℓ_p sensitivities are a straightforward generalization of the leverage scores. Given a matrix A and a row a of the matrix, the ℓ_p sensitivity of a w.r.t. the matrix A is defined as

$$\tau_A^{\ell_p}(a) := \max_{x : Ax \neq 0} \frac{|\langle a, x\rangle|^p}{\|Ax\|_p^p}.$$

The ℓ_p sensitivities measure the importance of a row to be able to estimate $\|Ax\|_p^p$ given any vector x. Suppose that a particular row a is orthogonal to all the other rows of the matrix A, we can see that a is very important to be able to approximate $\|Ax\|_p^p$ up to a multiplicative factor. It can be shown that if the matrix A has d columns, then the sum of ℓ_p sensitivities $\sum_{a \in A} \tau_A^{\ell_p}(a) \le d^{\max(p/2,1)}$ [MMWY22]. Now we state the following sampling result which shows that sampling rows of the matrix A with probabilities depending on the sensitivities and appropriately rescaling the sampled rows gives an ℓ_p subspace embedding.

Theorem 14.3.1. *Given a matrix A and a vector $v \in [0,1]^n$ such that for all $i \in [n]$, $v_i \ge \beta \tau_A^{\ell_p}(a_i)$ for some $\beta \le 1$, let a random diagonal matrix $\mathbf{S}$ be generated as follows: for each $i \in [n]$ independently, set $\mathbf{S}_{ii} = (1/p_i)^{1/p}$ with probability p_i and 0 otherwise. If $p_i \ge \min(1, C_1 \beta^{-1} v_i (C_2 d \log(d/\varepsilon) + \log(1/\delta))/\varepsilon^2)$ for large enough constants C_1 and C_2, then with probability $\ge 1 - \delta$, for all $x \in \mathbb{R}^d$,*

$$\|SAx\|_p = (1 \pm \varepsilon)\|Ax\|_p.$$

Given constant factor approximations for the ℓ_p sensitivities we can define the probabilities p_i such that the matrix S has at most $O(d^{\max(p/2,1)}(d \log(d/\varepsilon) + \log 1/\delta)/\varepsilon^2)$ non-zero entries with a large probability. The proof of the above theorem proceeds by showing that for a fixed vector x, the event $\|SAx\|_p = (1 \pm \varepsilon)\|Ax\|_p$ holds with a high probability and then using an ε-net argument to extend the high probability guarantee for a single vector x to a guarantee for all the vectors x. For $p = 2$, we can show that in the above theorem $p_i \ge \min(1, C_1 \beta^{-1} v_i (C_2 \log(d/\varepsilon) + \log 1/\delta)\varepsilon^{-2})$ suffices to construct a subspace embedding. So, in all our results for the special case of 2, only $\widetilde{O}(d)$ rows need to be sampled.

We will now show a construction of a composable sketch $\text{sk}(\mathscr{A})$ given a dataset $\mathscr{A}$. The composable sketch $\text{sk}(\mathscr{A})$ can be used to construct an ℓ_p subspace embedding for the matrix $\mathscr{A}.\text{vals}$. Importantly, we note that given composable sketches $\text{sk}(\mathscr{A})$ and $\text{sk}(\mathscr{B})$, the sketches can be merged only when $\mathscr{A}$ and $\mathscr{B}$ are conforming and the sketches $\text{sk}(\mathscr{A})$ and $\text{sk}(\mathscr{B})$ are constructed using the same randomness in a way which will become clear after we give the sketch construction.

We parameterize our sketch construction with an integer parameter t that defines the number of times a sketch can be merged with other sketches. We denote the sketch by $\text{sk}_t(\mathscr{A})$ if it is "mergeable" t times. Merging $\text{sk}_t(\mathscr{A})$ and $\text{sk}_{t'}(\mathscr{A})$ gives $\text{sk}_{\min(t,t')-1}(\mathscr{A} \cup \mathscr{B})$. Naturally, the size of the sketch increases with the parameter t. We will first show how the sketch $\text{sk}_t(\mathscr{A})$ is created.

14.3.2 Sketch Creation

Given a dataset $\mathscr{A}$ and a parameter t, we pick t independent *fully random* hash functions $h_1, \dots, h_t$ mapping keys to uniform random variables in the interval $[0,1]$. Thu, for each key, the value $h_i(\text{key})$ is an independent uniform random variable in the interval $[0,1]$. Given hash functions $h_1, h_2, \dots, h_t$, first for each (key, val) $\in \mathscr{A}$, we compute $\widetilde{\tau}_{\text{key}}$ that satisfies

$$(1+\varepsilon)^t \tau_{\mathscr{A}.\text{vals}}^{\ell_p}(\text{val}) \le \widetilde{\tau}_{\text{key}} \le (1+\varepsilon)^{t+1} \tau_{\mathscr{A}.\text{vals}}^{\ell_p}(\text{val}).$$

Note that we are free to choose $\widetilde{\tau}_{\text{key}}$ to be any value in the above interval. To allow randomness in computing the values of $\widetilde{\tau}_{\text{key}}$, we introduce another parameter γ. We assume that with probability $1-\gamma$, for all key $\in \mathscr{A}$.keys, $\widetilde{\tau}_{\text{key}}$ satisfies the above relation. When creating the sketch from scratch, as in the dataset $\mathscr{A}$ is entirely available, since we can compute exact ℓ_p sensitivities, we can take γ to be 0. The only requirement is that the value of $\widetilde{\tau}_{\text{key}}$ must be computed independently of the hash functions $h_1, \ldots, h_t$.

For each key $\in \mathscr{A}$.keys, let $p_{\text{key}} = C\widetilde{\tau}_{\text{key}}(d \log d/\varepsilon + \log 1/\delta)\varepsilon^{-2}$ and now for each h_i, define

$$\mathsf{senSample}(\mathscr{A}, h_i, t) \coloneqq \{(\text{key}, \text{val}, \min(p_{\text{key}}, 1)) \mid (\text{key}, \text{val}) \in \mathscr{A}, h_i(\text{key}) \le p_{\text{key}}\}.$$

The sketch $\text{sk}_{t,0}(\mathscr{A})$ is now defined to be the collection $(\mathsf{senSample}(\mathscr{A}, h_1, t), \ldots, \mathsf{senSample}(\mathscr{A}, h_t, t))$. The procedure is described in Algorithm 14.1. Note that for each $(\text{key}, \text{val}) \in \mathscr{A}$,

$$\begin{aligned}\mathbf{Pr}_{h_i}[(\text{key}, \text{val}, *) \in \mathsf{senSample}(\mathscr{A}, h_i, t)] &= \min(p_{\text{key}}, 1)\\ &\ge \min(C\tau^{\ell_p}_{\mathscr{A}.\text{vals}}(\text{val})(d\log(d/\varepsilon) + \log 1/\delta)\varepsilon^{-2}, 1).\end{aligned}$$

Hence, the construction of the set $\mathsf{senSample}(\mathscr{A}, h_i, t)$ is essentially performing ℓ_p sensitivity sampling as in Theorem 14.3.1 and for sampled rows it also stores the probability with which they were sampled. Thus, a matrix constructed appropriately using $\mathsf{senSample}(\mathscr{A}, h_i, t)$ will be a subspace embedding for the matrix $\mathscr{A}$.vals with probability $\ge 1-\delta$.

Throughout the construction, we ensure that the sketch $\text{sk}_{t,\gamma}(\mathscr{A}) = (\mathsf{senSample}(\mathscr{A}, h_1, t), \ldots, \mathsf{senSample}(\mathscr{A}, h_t, t))$ satisfies the following definition.

Definition 14.3.2. A sketch $(\mathsf{senSample}(\mathscr{A}, h_1, t), \ldots, \mathsf{senSample}(\mathscr{A}, h_t, t))$ is denoted $\text{sk}_{t,\gamma}(\mathscr{A})$ if with probability $\ge 1-\gamma$ (over randomness independent of $h_1, \ldots, h_t$), for each $(\text{key}, \text{val}) \in \mathscr{A}$, there exist values $\widetilde{\tau}_{\text{key}}$ (computed independently of the hash functions $h_1, \ldots, h_t$) such that

$$(1+\varepsilon)^t \tau^{\ell_p}_{\mathscr{A}.\text{vals}}(\text{val}) \le \widetilde{\tau}_{\text{key}} \le (1+\varepsilon)^{t+1} \tau^{\ell_p}_{\mathscr{A}.\text{vals}}(\text{val}) \tag{14.1}$$

and for $p_{\text{key}} = C\widetilde{\tau}_{\text{key}}(d \log d/\varepsilon + \log 1/\delta)\varepsilon^{-2}$,

$$\mathsf{senSample}(\mathscr{A}, h_i, t) = \{(\text{key}, \text{val}, \min(p_{\text{key}}, 1)) \mid (\text{key}, \text{val}) \in \mathscr{A}, h_i(\text{key}) \le p_{\text{key}}\}. \tag{14.2}$$

Note that using the bounds on the sum of ℓ_p sensitivities, we obtain that with probability $\ge 1-\gamma-\exp(-d)$, the size of the sketch $\text{sk}_{t,\gamma}(\mathscr{A})$, in terms of the number of rows, is $O(t(1+\varepsilon)^{t+1} d^{\max(p/2,1)}(d \log d/\varepsilon + \log 1/\delta)\varepsilon^{-2})$. By Theorem 14.3.1, we obtain that given a sketch $\text{sk}_{t,\gamma}(\mathscr{A})$, Algorithm 14.2 computes a subspace embedding for the matrix $\mathscr{A}$.vals. Thus, we have the following theorem.

Theorem 14.3.3. *Given $\text{sk}_{t,\gamma}(\mathscr{A})$ constructed with parameters ε, δ, Algorithm 14.2 returns a matrix that* *with probability* $\geq 1-\gamma-\delta$ *satisfies, for all* x*,*

$$\|Mx\|_p^p = (1 \pm \varepsilon)\|\mathscr{A}.\mathrm{vals} \cdot x\|_p^p.$$

We now show how to compute a sketch for $\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s$ given sketches $\mathrm{sk}_{t_1,\gamma_1}(\mathscr{A}_1), \ldots, \mathrm{sk}_{t_s,\gamma_s}(\mathscr{A}_s)$ for s conforming datasets $\mathscr{A}_1, \ldots, \mathscr{A}_s$.

14.3.3 Merging Sketches

Theorem 14.3.4. *Let* $\mathscr{A}_1, \ldots, \mathscr{A}_s$ *be conforming datasets. Given sketches* $\mathrm{sk}_{t_1,\gamma_1}(\mathscr{A}_1), \ldots, \mathrm{sk}_{t_s,\gamma_s}(\mathscr{A}_s)$ *constructed using the same hash functions* $h_1, \ldots$ *and parameters* $\varepsilon, \delta > 0$*, Algorithm 14.3 then computes a sketch* $\mathrm{sk}_{\min_i(t_i)-1,\delta+\gamma_1+\cdots+\gamma_s}(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s)$*.*

Proof Outline

In the original sketch creation procedure, we compute approximations to the ℓ_p sensitivities which we use to compute a value p_{key} and keep all the (key, val) pairs satisfying $h(\mathrm{key}) \leq p_{\mathrm{key}}$. We argued that the original sketch creation is essentially an implementation of the ℓ_p sensitivity sampling algorithm in Theorem 14.3.1. Now, given sketches of $\mathscr{A}_1, \ldots, \mathscr{A}_s$, we want to *simulate* the ℓ_p sensitivity sampling of the rows in the matrix $(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s)$.vals to create the sketch $\mathrm{sk}(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_k)$. An important property of the ℓ_p sensitivities is the *monotonicity* – the ℓ_p sensitivity of a row only goes down with adding new rows to the matrix. Suppose we have a way to compute ℓ_p sensitivities of the rows of the matrix $(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s)$.vals. Suppose a row $a \in \mathscr{A}_1$.vals. Then the probability that it has to be sampled when performing ℓ_p sensitivity sampling on the matrix $(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s)$.vals is smaller than the probability that the row has to be sampled when performing ℓ_p sensitivity sampling on the matrix $\mathscr{A}_1$.vals. Thus, the rows that we ignored when constructing $\mathrm{sk}(\mathscr{A}_1)$ "don't really matter" as the ℓ_p sensitivity sampling of the rows of $(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s)$.vals when performing using the same hash function h would also not have sampled that row since $h(\mathrm{key})$ was already larger than the probability that $\mathscr{A}_1$ assigned to the row a which is in turn larger than the probability that $\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_t$ assigned to the row a.

The above argument assumes that we have a way to approximate the ℓ_p sensitivity of a row with respect to the matrix $\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s$ and sensitivity sampling requires that these approximations be independent of the hash function h we are using to simulate sensitivity sampling. We now recall that each $\mathrm{sk}_{t,\gamma}(\mathscr{A})$ has t independent copies of the senSample data structure. We show that one of the copies can be used to compute approximate sensitivities and then perform the ℓ_p sensitivity sampling on the other copies. Thus, each time we merge a $\mathrm{sk}_{t,\gamma}(\cdot)$ data structure, we lose a copy of the senSample data structure in the sketch which is why the sketch $\mathrm{sk}_{t,\gamma}(\cdot)$ can be merged only t times in the future.

Formal Proof

Proof. Let $\mathscr{A} \coloneqq \mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s$ and $t = \min(t_1, \ldots, t_s)$. Recall that each $\mathrm{sk}_{t_j,\gamma_j}(\mathscr{A}_j)$ is a collection of the data structures $\mathsf{senSample}(\mathscr{A}_j, h_1, t_j), \ldots, \mathsf{senSample}(\mathscr{A}_j, h_{t_j}, t_j)$ and that by definition of $\mathrm{sk}_{t,\gamma}(\mathscr{A})$, for each $j = 1, \ldots, s$, with probability $1 - \gamma_j$ (over independent randomness $h_1, \ldots, h_{t_j}$) for each $(\mathrm{key}, \mathrm{val}) \in \mathscr{A}_j$, there exists $\widetilde{\tau}^{(j)}_{\mathrm{key}}$ for which

$$(1+\varepsilon)^{t_j} \tau^{\ell_p}_{\mathscr{A}_j.\mathrm{vals}}(\mathrm{val}) \le \widetilde{\tau}^{(j)}_{\mathrm{key}} \le (1+\varepsilon)^{t_j+1} \tau^{\ell_p}_{\mathscr{A}_j.\mathrm{vals}}(\mathrm{val}) \tag{14.3}$$

and for $p_{\mathrm{key}} = C\widetilde{\tau}^{(j)}_{\mathrm{key}}(d \log(d/\varepsilon) + \log 1/\delta)\varepsilon^{-2}$ and $i = 1, \ldots, t_j$,

$$\mathsf{senSample}(\mathscr{A}_j, h_i, t_j) = \{(\mathrm{key}, \mathrm{val}, \min(p_{\mathrm{key}}, 1)) \mid (\mathrm{key}, \mathrm{val}) \in \mathscr{A}, h_i(\mathrm{key}) \le p_{\mathrm{key}}\}.$$

By a union bound, with probability $\ge 1 - (\gamma_1 + \cdots + \gamma_s)$, we have $\widetilde{\tau}^{(j)}_{\mathrm{key}}$ as in (14.3) for all $j = 1, \ldots, s$ and $\mathrm{key} \in \mathscr{A}_j.\mathrm{keys}$. Condition on this event.

We now show that the matrix M constructed by the algorithm is a subspace embedding for $(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s).\mathrm{vals}$. Note that, in constructing the matrix M, the algorithm uses $\mathsf{senSample}$ data structures all constructed using the same hash function h_t.

If $(\mathrm{key}, \mathrm{val}, *)$ is in *any* of the sets $\mathsf{senSample}(\mathscr{A}_1, h_t, t_1), \ldots, \mathsf{senSample}(\mathscr{A}_s, h_t, t_s)$, let $p^{\mathrm{merge}}_{\mathrm{key}}$ be the maximum "probability value" among all the tuples with $(\mathrm{key}, \mathrm{val}, *)$. Let S be the set formed by all the tuples $(\mathrm{key}, \mathrm{val}, p^{\mathrm{merge}}_{\mathrm{key}})$. For each $(\mathrm{key}, \mathrm{val}) \in \mathscr{A}$, define

$$\widetilde{\tau}^{\mathrm{merge}}_{\mathrm{key}} = \max_{(\mathrm{key},\mathrm{val}) \in \mathscr{A}_j} \widetilde{\tau}^{(j)}_{\mathrm{key}}.$$

Now, for each $(\mathrm{key}, \mathrm{val}) \in \mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s$,

$$\begin{aligned}\mathbf{Pr}[(\mathrm{key}, \mathrm{val}, *) \in S] &= \mathbf{Pr}[h_t(\mathrm{key}) \le \max_{j:(\mathrm{key},\mathrm{val}) \in \mathscr{A}_j} C\widetilde{\tau}^{\mathrm{merge}}_{\mathrm{key}}(d \log d/\varepsilon + \log 1/\delta)\varepsilon^{-2}] \\ &= p^{\mathrm{merge}}_{\mathrm{key}}.\end{aligned}$$

By monotonicity of ℓ_p sensitivities, if $(\mathrm{key}, \mathrm{val}) \in \mathscr{A}_j$, then

$$\tau^{\ell_p}_{(\mathscr{A}_1 \cup \cdots \cup \mathscr{A}_s).\mathrm{vals}}(\mathrm{val}) \le \tau^{\ell_p}_{\mathscr{A}_j.\mathrm{vals}}(\mathrm{val}) \le \widetilde{\tau}^{(j)}_{\mathrm{key}} \le \widetilde{\tau}^{\mathrm{merge}}_{\mathrm{key}}.$$

Hence, with probability $\ge 1 - \delta$ the set S is a leverage score sample of the rows of the matrix $\mathscr{A}.\mathrm{vals}$. By a union bound, with probability $\ge 1 - (\delta + \gamma_1 + \cdots + \gamma_s)$, the matrix M with rows given by $1/(p^{\mathrm{merge}}_{\mathrm{key}})^{1/p} \cdot \mathrm{val}$ for $(\mathrm{key}, \mathrm{val}, p^{\mathrm{merge}}_{\mathrm{key}}) \in S$ is an ℓ_p subspace embedding for the matrix $\mathscr{A}.\mathrm{vals}$ and satisfies for all x,

$$\|Mx\|_p^p = (1 \pm \varepsilon/4)\|\mathscr{A}.\mathrm{vals} \cdot x\|_p^p.$$

For each $(\mathrm{key}, \mathrm{val}) \in \mathscr{A}$, we can compute

$$\widetilde{\tau}_{\mathrm{key}}^{\mathrm{approx}} = (1+\varepsilon)^{t-1}(1+\varepsilon/4)\max_x \frac{|\langle \mathrm{val}, x\rangle|^p}{\|Mx\|_p^p}.$$

Conditioned on M being a subspace embedding for $\mathscr{A}$, we have that $(1+\varepsilon)^{t-1}\tau_{\mathscr{A}.\mathrm{vals}}^{\ell_p}(\mathrm{val}) \le \widetilde{\tau}_{\mathrm{key}}^{\mathrm{approx}} \le (1+\varepsilon)^t \tau_{\mathscr{A}.\mathrm{vals}}^{\ell_p}(\mathrm{val})$. For each $(\mathrm{key}, \mathrm{val}) \in \mathscr{A}_j$, we have

$$\widetilde{\tau}_{\mathrm{key}}^{\mathrm{approx}} \le (1+\varepsilon)^t \tau_{\mathscr{A}.\mathrm{vals}}^{\ell_p}(\mathrm{val}) \le (1+\varepsilon)^t \tau_{\mathscr{A}_j.\mathrm{vals}}^{\ell_p}(\mathrm{val}) \le \widetilde{\tau}_{\mathrm{key}}^{(j)} \le \widetilde{\tau}_{\mathrm{key}}^{\mathrm{merge}}. \tag{14.4}$$

Thus, with probability $\ge 1 - (\delta + \gamma_1 + \cdots + \gamma_s)$, for all $(\mathrm{key}, \mathrm{val}) \in \mathscr{A}$,

$$(1+\varepsilon)^{t-1}\tau_{\mathscr{A}.\mathrm{vals}}^{\ell_p}(\mathrm{val}) \le \widetilde{\tau}_{\mathrm{key}}^{\mathrm{approx}} \le (1+\varepsilon)^t \tau_{\mathscr{A}.\mathrm{vals}}^{\ell_p}(\mathrm{val}).$$

Now for all $i \le t-1$, we define $\widetilde{p}_{\mathrm{key}} = C\widetilde{\tau}_{\mathrm{key}}^{\mathrm{approx}}(d \log d/\varepsilon + \log 1/\delta)\varepsilon^{-2}$ and

$$\mathsf{senSample}(\mathscr{A}, h_i, t-1) = \{(\mathrm{key}, \mathrm{val}, \min(1, \widetilde{p}_{\mathrm{key}})) \mid (\mathrm{key}, \mathrm{val}) \in \mathscr{A}, h_i(\mathrm{key}) \le \widetilde{p}_{\mathrm{key}}\}$$

and have

$$\mathbf{Pr}_{h_i}[(\mathrm{key}, \mathrm{val}, *) \in \mathsf{senSample}(\mathscr{A}, h_i, t-1)] = \min(1, \widetilde{p}_{\mathrm{key}}).$$

Note that while the above definition says to construct the set by looking at each $(\mathrm{key}, \mathrm{val}) \in \mathscr{A}$, as $\widetilde{\tau}_{\mathrm{key}}^{\mathrm{approx}} \le \max_{j:(\mathrm{key},\mathrm{val})\in\mathscr{A}_j} \widetilde{\tau}_{\mathrm{key}}^{(j)}$ by definition, we only have to look at the elements of the set $\mathsf{senSample}(\mathscr{A}_1, h_i, t-1), \ldots, \mathsf{senSample}(\mathscr{A}_s, h_i, t-1)$ as all other missing elements from $\mathscr{A}$ would not have been included in the set anyway. Here the property that the $\widetilde{\tau}$ values satisfy (14.3) becomes crucial.

Thus, we have that the algorithm constructs $\mathrm{sk}_{t-1,\delta+\gamma_1+\cdots+\gamma_s}(\mathscr{A})$. □

14.3.4 Neighborhood Propagation

As described in the previous section, the neighborhood propagation algorithm using the composable sketches lets each node compute a subspace embedding for the matrix formed by the data of the matrices in a neighborhood around the node. We will now analyze the setting of the δ parameter in the ℓ_p composable sketch construction.

We have that merging the sketches $\mathrm{sk}_{t_1,\gamma_1}(\mathscr{A}_1), \ldots, \mathrm{sk}_{t_s,\gamma_s}(\mathscr{A}_s)$, we obtain $\mathrm{sk}_{\min_i t_i - 1, \delta+\gamma_1+\cdots+\gamma_s}(\mathscr{A}_1 \cup$

$\cdots \cup \mathscr{A}_s)$. Let s be the total number of nodes in the graph. The sketches that each neighborhood obtains are merged at most Δ times. Hence, setting $\delta = \delta'/(2s)^{\Delta}$, each node in the graph computes a sketch for the data in its neighborhood with the probability parameter δ'. Further, setting $\delta' = 1/10s$, we obtain by a union bound that with probability $\geq 9/10$, all the nodes in the graph compute an ℓ_p subspace embeddings for the data in their Δ neighborhoods. Thus, we have the following theorem.

Theorem 14.3.5. *Suppose $G = (V, E)$ is an arbitrary graph with $|V| = s$. Each node in the graph knows and can communicate only with its neighbors. Given a distance parameter Δ and accuracy parameter $\varepsilon < 1/\Delta$, there is a neighborhood propagation algorithm that runs for Δ rounds such that at the end of the algorithm, with probability $\geq 9/10$, each vertex u in the graph computes an ε ℓ_p subspace embedding for the matrix formed by the data in the Δ neighborhood of u.*

In each of the Δ rounds, each node communicates at most $O(\Delta \cdot d^{\max(p/2,1)}(d \log d + \Delta \log s)\varepsilon^{-2})$ rows along with additional information for each row to all its neighbors. For $p = 2$, each node communicates $O(\Delta \cdot d(\log d + \Delta \log s)\varepsilon^{-2})$ rows to each of its neighbors in each round.

Since in many problems of interest, the parameter Δ is a small constant, the algorithm is communication efficient.

Algorithm 14.1: Creating the sketch $\text{sk}_{t,0}$ given $\mathscr{A}$

```
Input: A dataset $\mathscr{A}$ of pairs (key, val), an integer parameter $t \geq 1, \varepsilon, \delta$
Output: A sketch $\text{sk}_{t,0}(\mathscr{A})$
1  Let $h_1, \ldots, h_t$ be independent fully random hash functions with $h_i(\text{key})$ being a uniform
     random variable from $[0, 1]$;
2  For each $(\text{key}, \text{val}) \in \mathscr{A}$, $\tau^{\ell_p}_{\mathscr{A}.\text{vals}}(\text{val}) \leftarrow \max_x |\langle \text{val}, x\rangle|^p / \|\mathscr{A}.\text{vals} \cdot x\|_p^p$;
3  For each $(\text{key}, \text{val}) \in \mathscr{A}$, $p_{\text{key}} \leftarrow C\tau^{\ell_p}(d \log d + \log 1/\delta)\varepsilon^{-2}$;
4  for $i = 1, \ldots, t$ do
5      senSample$(\mathscr{A}, h_i, t) \leftarrow \emptyset$;
6      for $(\text{key}, \text{val}) \in \mathscr{A}$ do
7          if $h_i(\text{key}) \leq p_{\text{key}}$ then
8              senSample$(\mathscr{A}, h_i, t) \leftarrow$ senSample$(\mathscr{A}, h_i, t) \cup \{ (\text{key}, \text{val}, \min(1, p_{\text{key}})) \}$;
9          end
10     end
11 end
12 $\text{sk}_{t,0}(\mathscr{A}) \leftarrow$ (senSample$(\mathscr{A}, h_1, t), \ldots,$ senSample$(\mathscr{A}, h_t, t)$);
```

14.3.5 Applications to ℓ_p Regression

Let $\mathscr{A}$ be a dataset. In a (key, val) pair with val being a d dimensional vector, we treat the first $d - 1$ coordinates as the features and the last coordinate as the label. Then the ℓ_p linear regression problem

Algorithm 14.2: Computing a subspace embedding from a sketch

```
Input: Sketch $\mathrm{sk}_{t,\gamma}(\mathcal{A})$ constructed with parameters $\varepsilon, \delta$
Output: A matrix $M$ that is an $\varepsilon$ $\ell_p$ subspace embedding
1 Note $\mathrm{sk}_{t,\gamma}(\mathcal{A}) = (\mathsf{senSample}(\mathcal{A}), h_1, t), \ldots, \mathsf{senSample}(\mathcal{A}), h_t, t))$;
2 $M \leftarrow$ matrix with rows given by $(1/p_{\mathrm{key}})^{1/p} \cdot \mathrm{val}$ for
    $(\mathrm{key}, \mathrm{val}, p_{\mathrm{key}}) \in \mathsf{senSample}(\mathcal{A}, h_1, t)$;
3 return $M$
```

Algorithm 14.3: Merging Sketches

```
Input: Sketches $\mathrm{sk}_{t_1,\gamma_1}(\mathcal{A}_1), \ldots, \mathrm{sk}_{t_s,\gamma_s}(\mathcal{A}_s)$ constructed with the same parameters $\varepsilon, \delta$ and
        the same hash functions $h_1, \ldots,$
Output: Sketch $\mathrm{sk}_{\min_i t_i - 1, \delta + \sum_i \gamma_i}(\mathcal{A}_1 \cup \cdots \mathcal{A}_k)$
1 Let $h_1, h_2, \ldots,$ be the hash functions used in the construction of the sketches;
2 $t \leftarrow \min_i t_i$;
3 $\mathcal{A} \leftarrow \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_s$;                                   // Only notational
4 merge $\leftarrow \{(\mathrm{key}, \mathrm{val}) \mid \exists j \in [s], (\mathrm{key}, \mathrm{val}, *) \in \mathsf{senSample}(\mathcal{A}_j, h_t, t_j)\}$;
5 For each $(\mathrm{key}, \mathrm{val}) \in$ merge, $p^{\mathrm{merge}}_{\mathrm{key}} \leftarrow \max p$ with $(\mathrm{key}, \mathrm{val}, p) \in \cup_j \mathsf{senSample}(\mathcal{A}_j, h_t, t_j)$;
6 $M \leftarrow$ matrix with rows given by $(1/p^{\mathrm{merge}}_{\mathrm{key}})^{1/p} \cdot \mathrm{val}$ for $(\mathrm{val}, \mathrm{key}) \in$ merge;
7 for $i = 1, \ldots, t-1$ do
8     $\mathsf{senSample}(\mathcal{A}, h_i, t-1) \leftarrow \emptyset$;
9     $\mathrm{merge}_i \leftarrow \{(\mathrm{key}, \mathrm{val}) \mid \exists j \in [s], (\mathrm{key}, \mathrm{val}, *) \in \mathsf{senSample}(\mathcal{A}_j, h_i, t_j)\}$;
10    For each $(\mathrm{key}, \mathrm{val}) \in \mathrm{merge}_i$, $p^{(i)}_{\mathrm{key}} \leftarrow \max p$ with
        $(\mathrm{key}, \mathrm{val}, p) \in \cup_j \mathsf{senSample}(\mathcal{A}_j, h_i, t_j)$;
11    for $(\mathrm{key}, \mathrm{val}) \in \mathrm{merge}_i$ do
12        $\widetilde{\tau}^{\mathrm{approx}}_{\mathrm{key}} \leftarrow (1+\varepsilon)^{t-1}(1+\varepsilon/4)\max_x \frac{|\langle \mathrm{val}, x\rangle|^p}{\|Mx\|_p^p}$;
13        $p_{\mathrm{key}} \leftarrow C\widetilde{\tau}^{\mathrm{approx}}_{\mathrm{key}}(d \log d/\varepsilon + \log 1/\delta)\varepsilon^{-2}$;
14        if $\min(1, p_{\mathrm{key}}) > p^{(i)}_{\mathrm{key}}$ then
15            Output FAIL;
16        end
17        if $h_i(\mathrm{key}) \leq p_{\mathrm{key}}$ then
18            $\mathsf{senSample}(\mathcal{A}, h_i, t-1) \leftarrow \mathsf{senSample}(\mathcal{A}, h_i, t-1) \cup \{(\mathrm{key}, \mathrm{val}, \min(1, p_{\mathrm{key}}))\}$;
19        end
20    end
21 end
22 $\mathrm{sk}_{t-1, \delta+\gamma_1+\cdots+\gamma_s}(\mathcal{A}) \leftarrow (\mathsf{senSample}(\mathcal{A}, h_1, t-1), \ldots, \mathsf{senSample}(\mathcal{A}, h_{t-1}, t-1))$;
```

on a dataset $\mathscr{A}$ is

$$\min_{x \in \mathbb{R}^{d-1}} \|\mathscr{A}.\mathrm{vals}\begin{bmatrix} x \\ -1 \end{bmatrix}\|_p^p.$$

Thus, if the matrix M is an ε subspace embedding for the matrix $\mathscr{A}$.vals, then

$$\widetilde{x} = \arg\min_x \|M\begin{bmatrix} x \\ -1 \end{bmatrix}\|_p^p,$$

then

$$\|\mathscr{A}.\mathrm{vals}\begin{bmatrix} \widetilde{x} \\ -1 \end{bmatrix}\|_p^p \le (1 + O(\varepsilon)) \min_x \|\mathscr{A}.\mathrm{vals}\begin{bmatrix} x \\ -1 \end{bmatrix}\|_p.$$

Thus, composable sketches for constructing ℓ_p subspace embeddings can be used to solve ℓ_p regression problems.

14.3.6 Low Rank Approximation

We consider the Frobenius norm low rank approximation. Given a matrix A, a rank parameter k we want to compute a rank k matrix B such that $\|A - B\|_F^2$ is minimized. The optimal solution to this problem can be obtained by truncating the singular value decomposition of the matrix A to its top k singular values. As computing the exact singular value decomposition of a matrix A is slow, the approximate version of low rank approximation has been heavily studied in the literature [CW17]. In the approximate version, given a parameter ε, we want to compute a rank-k matrix B such that

$$\|A - B\|_F^2 \le (1 + \varepsilon) \min_{\text{rank-}k\ B} \|A - B\|_F^2.$$

As the number of rows in A is usually quite large, the version of the problem which asks to only output a k dimensional subspace V of $\mathbb{R}^d$ is also studied:

$$\|A(I - \mathbb{P}_V)\|_F^2 \le (1 + \varepsilon) \min_{\text{rank-}k\ B} \|A - B\|_F^2.$$

Here $\mathbb{P}_V$ denotes the orthogonal projection matrix onto the subspace V.

We show that using composable sketches for ℓ_2 sensitivity sampling, we can solve the low rank approximation problem. While the composable sketch for ℓ_2 sensitivity sampling has $\widetilde{O}(d)$ rows, we will show that for solving the low rank approximation problem, the composable sketch need only have $\widetilde{O}(k)$ rows. We use the following result.

Theorem 14.3.6 ([CW09, Theorem 4.2])**.** *If A is an $n \times d$ matrix and $\boldsymbol{R}$ is a $d \times m$ random sign matrix for*

$m = O(k\log(1/\delta)/\varepsilon)$*, then with probability* $\geq 1-\delta$,

$$\min_{\text{rank-}k\,X} \|A\boldsymbol{R}X - A\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \min_{\text{rank-}k\,B} \|A-B\|_{\mathrm{F}}^2.$$

Using the affine embedding result of [CW17], if $\boldsymbol{L}$ is now a leverage score sampling matrix, meaning that $\boldsymbol{L}$ is a diagonal matrix with the entry $1/\sqrt{p_i}$ for the rows that are sampled by the leverage score sampling algorithm as in Theorem 14.3.1, then for all matrices X

$$\|\boldsymbol{L}A\boldsymbol{R}X - \boldsymbol{L}A\|_{\mathrm{F}}^2 = (1\pm\varepsilon)\|A\boldsymbol{R}X - A\|_{\mathrm{F}}^2.$$

Hence, if

$$\widetilde{X} = \arg\min_{\text{rank-}k\,X} \|\boldsymbol{L}A\boldsymbol{R}X - \boldsymbol{L}A\|_{\mathrm{F}}^2,$$

then $\|A\boldsymbol{R}\widetilde{X} - A\|_{\mathrm{F}}^2 \leq (1+O(\varepsilon))\min_{\text{rank-}k\,B}\|A-B\|_{\mathrm{F}}^2$ which implies that $\|A(I - \mathbb{P}_{\text{rowspace}(\boldsymbol{R}\widetilde{X})})\|_{\mathrm{F}}^2 \leq (1+O(\varepsilon))\min_{\text{rank-k}\,B}\|A-B\|_{\mathrm{F}}^2$.

Thus, if $\boldsymbol{R}$ is a random sign matrix with $O(k\log(1/\delta)/\varepsilon)$ rows, then $\mathrm{sk}_{t,\gamma}(\mathscr{A}.\mathrm{vals}\cdot\boldsymbol{R})$, along with the corresponding rows in $\mathscr{A}.\mathrm{vals}$ for the rows in $\mathrm{sk}_{t,\gamma}(\mathscr{A}.\mathrm{vals}\cdot\boldsymbol{R})$, can be used to compute a $1+\varepsilon$ approximation to the low rank approximation problem. As the matrix $\mathscr{A}.\mathrm{vals}\cdot\boldsymbol{R}$ has only $\widetilde{O}(k)$ rows, the composable sketch $\mathrm{sk}_{t,\gamma}(\mathscr{A}.\mathrm{vals}\cdot\boldsymbol{R})$ has a number of rows that depends only on k as well.

14.4 Conclusions and Open Questions

In this work, we introduce the personalized CONGEST model and obtain communication efficient algorithms for computing ℓ_p subspace embeddings and the Frobenius norm low rank approximation problem. Our protocol heavily utilizes the monotonicity of ℓ_p sensitivities and therefore for $p \neq 2$ the subspace embeddings are suboptimal by a factor d compared to the subspace embeddings constructed using Lewis weights [Lew78, CP15]. An interesting question to study is if we can obtain communication bounds with the optimal dependence in d for $p \neq 2$.

To decouple the randomness in probability computation from randomness in sampling, our protocol in the first round essentially communicates information to compute Δ independent subspace embeddings.e It is an interesting open question to obtain protocols that do not need to decouple the randomness in this way and do not suffer the multiplicative Δ factor in the amount of communication required.

Chapter 15

Bibliography

[AC09] Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009. 6, 8

[ACK$^+$16] Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P. Woodruff, and Qin Zhang. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 311–319, 2016. 135

[ACW17] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM J. Matrix Anal. Appl.*, 38(4):1116–1138, 2017. 134, 137

[ADF$^+$23] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023. 173

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: Sparsification, Spanners, and Subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 5–14, 2012. 193

[AIV19] Anders Aamand, Piotr Indyk, and Ali Vakilian. (learned) frequency estimation algorithms under zipfian distribution. *arXiv preprint arXiv:1908.05198*, 2019. 177, 184

[AKK$^+$20] Thomas D Ahle, Michael Kapralov, Jakob BT Knudsen, Rasmus Pagh, Ameya Velingker, David Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 141–160. SIAM, 2020. 14, 151, 159, 160, 176, 181, 182, 184

[AKO11] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 363–372. IEEE, 2011. 198, 199

[AKO18] Mohamad Ahmadi, Fabian Kuhn, and Rotem Oshman. Distributed approximate maximum matching in the CONGEST model. In *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 339

[ALO15] Zeyuan Allen Zhu, Zhenyu Liao, and Lorenzo Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 237–245, 2015. 134

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999. 3, 298

[And17] Alexandr Andoni. High frequency moments via max-stability. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6364–6368. IEEE, 2017. 6, 15, 16, 198, 203, 216, 217, 224

[ANPW13] Alexandr Andoni, Huy L Nguyen, Yury Polyanskiy, and Yihong Wu. Tight lower bound for linear sketches of moments. In *International Colloquium on Automata, Languages, and Programming*, pages 25–32. Springer, 2013. 198

[ANW14] Haim Avron, Huy Nguyen, and David Woodruff. Subspace embeddings for the polynomial kernel. *Advances in neural information processing systems*, 27, 2014. 176

[AR20] Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 342–353. IEEE, 2020. 151

[AS15] Pankaj K Agarwal and R Sharathkumar. Streaming algorithms for extent problems in high dimensions. *Algorithmica*, 72(1):83–98, 2015. 17, 255, 271

[AS23a] Josh Alman and Zhao Song. Fast attention requires bounded entries. *arXiv preprint arXiv:2302.13214*, 2023. 13, 175

[AS23b] Sepehr Assadi and Janani Sundaresan. (Noisy) gap cycle counting strikes back: Random order streaming lower bounds for connected components and beyond. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 183–195, 2023. 275

[AZL16] Zeyuan Allen-Zhu and Yuanzhi Li. LazySVD: Even faster SVD decomposition yet without agonizing pain. *Advances in neural information processing systems*, 29, 2016. 132

[AZL17] Zeyuan Allen-Zhu and Yuanzhi Li. First efficient convergence for streaming k-PCA: a global, gap-free, and near-optimal rate. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 487–492. IEEE, 2017. 274

[BCIW16] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, and David P. Woodruff. Beating CountSketch for heavy hitters in insertion streams. In *STOC'16—Proceedings of the 48th*

Annual ACM SIGACT Symposium on Theory of Computing, pages 740–753. ACM, New York, 2016. 201

[BCW22] Ainesh Bakshi, Kenneth L. Clarkson, and David P. Woodruff. Low-rank approximation with $1/\varepsilon^{1/3}$ matrix-vector products. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, 2022. 112

[BDM⁺20] Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 517–528. IEEE Computer Soc., Los Alamitos, CA, 2020. 258, 261, 423, 424

[BDN15] Jean Bourgain, Sjoerd Dirksen, and Jelani Nelson. Toward a unified theory of sparse dimensionality reduction in Euclidean space. *Geometric and Functional Analysis*, 25(4):1009–1088, 2015. 30

[BDWY16] Maria-Florina Balcan, Simon Shaolei Du, Yining Wang, and Adams Wei Yu. An improved gap-dependency analysis of the noisy power method. In *Conference on Learning Theory*, pages 284–309. PMLR, 2016. 123, 274

[BEO⁺13] Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. A tight bound for set disjointness in the message-passing model. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*. IEEE Computer Society, 2013. 297

[BEPS16] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM (JACM)*, 63(3):1–45, 2016. 339

[BFD⁺22] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. Hydra attention: Efficient attention with many heads. In *European Conference on Computer Vision*, pages 35–49. Springer, 2022. 175

[BGKS06] Lakshminath Bhuvanagiri, Sumit Ganguly, Deepanjan Kesh, and Chandan Saha. Simpler algorithm for estimating frequency moments of data streams. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 708–713. ACM Press, 2006. 198, 199

[BGW20] Mark Braverman, Sumegha Garg, and David P. Woodruff. The coin problem with applications to data streams. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science*, pages 318–329. IEEE Computer Soc., Los Alamitos, CA, 2020. 198, 201

[BHPI02] Mihai Bādoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 250–257. ACM, 2002. 62

[BHSW20] Mark Braverman, Elad Hazan, Max Simchowitz, and Blake Woodworth. The gradient complexity of linear regression. In *Conference on Learning Theory*, pages 627–647. PMLR, 2020. 109

[BJKS04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004. 3, 298

[BJWY22] Omri Ben-Eliezer, Rajesh Jayaram, David Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *J. ACM*, 69(2):17:1–17:33, 2022. 343

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 180

[BKLW14] Maria-Florina Balcan, Vandana Kanchanapally, Yingyu Liang, and David Woodruff. Improved distributed principal component analysis. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3113–3121, 2014. 340

[BL94] Cajo JF Ter Braak and Caspar WN Looman. Biplots in reduced-rank regression. *Biometrical journal*, 36(8):983–1003, 1994. 119

[Ble90] Guy E Blelloch. Prefix sums and their applications. 1990. 186

[BLS$^+$15] Maria-Florina Balcan, Yingyu Liang, Le Song, David Woodruff, and Bo Xie. Distributed kernel principal component analysis. *arXiv preprint arXiv:1503.06858*, 2015. 340

[BMD$^+$23] Francesca Babiloni, Ioannis Marras, Jiankang Deng, Filippos Kokkinos, Matteo Maggioni, Grigorios Chrysos, Philip Torr, and Stefanos Zafeiriou. Linear complexity self-attention with 3rd order polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 175

[BMR$^+$20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 173

[BN23] Ainesh Bakshi and Shyam Narayanan. Krylov methods are (nearly) optimal for low-rank approximation. *arXiv preprint arXiv:2304.03191*, 2023. 91, 118, 121

[BO10] Vladimir Braverman and Rafail Ostrovsky. Zero-one frequency laws. In *STOC'10—Proceedings of the 2010 ACM International Symposium on Theory of Computing*, pages 281–290. ACM, New York, 2010. 199, 298

[Bou11] Christos Boutsidis. *Topics in Matrix Sampling Algorithms*. PhD thesis, Rensselaer Polytechnic Institute, USA, 2011. 120

[BPC20] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 174

[BS80] Jon Louis Bentley and James B Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980. 7

[BSS12] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012. 134, 137

[BW11] Joshua Brody and David P Woodruff. Streaming algorithms with one-sided estimation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 436–447. Springer, 2011. 151

[BW17] Christos Boutsidis and David P Woodruff. Optimal CUR matrix decompositions. *SIAM Journal on Computing*, 46(2):543–589, 2017. 34, 56, 58, 59

[BWZ16] Christos Boutsidis, David Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 236–249, 2016. 274, 340

[BYJKS04] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004. 15, 198

[BZB$^+$20] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *AAAI*, pages 7432–7439. AAAI Press, 2020. 189

[Car17] James B. Carrell. *Groups, Matrices, and Vector Spaces: A Group Theoretic Approach to Linear Algebra*. Springer New York, New York, NY, 2017. 390

[CASS21] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 169–182, 2021. 7

[CCF04] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004. 194, 200, 205, 208, 225, 236, 238, 246

[CCFC02] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*. Springer, 2002. 3, 6

[CCKW22] Nadiia Chepurko, Kenneth L Clarkson, Praneeth Kacham, and David P Woodruff. Near-optimal algorithms for linear algebra in the current matrix multiplication time. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3043–3068. SIAM, 2022. 20

[CCM08] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the fortieth annual ACM sympo-*

sium on Theory of computing, pages 641–650, 2008. 275

[CD13] Emmanuel J. Candès and Mark A. Davenport. How well can we estimate a sparse vector? *Appl. Comput. Harmon. Anal.*, 34(2):317–323, 2013. 81

[CDDR23] Shabarish Chenakkod, Michał Dereziński, Xiaoyu Dong, and Mark Rudelson. Optimal embedding dimension for sparse subspace embeddings. *arXiv preprint arXiv:2311.10680*, 2023. 8, 59, 60

[CEK$^+$15] Flavio Chierichetti, Alessandro Epasto, Ravi Kumar, Silvio Lattanzi, and Vahab Mirrokni. Efficient algorithms for public-private social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 139–148, 2015. 340

[CEM$^+$15] Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172. ACM, 2015. 30, 56, 62

[CGK$^+$17] Flavio Chierichetti, Sreenivas Gollapudi, Ravi Kumar, Silvio Lattanzi, Rina Panigrahy, and David P Woodruff. Algorithms for ℓ_p low-rank approximation. In *International Conference on Machine Learning*, pages 806–814. PMLR, 2017. 257

[CGRS19] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 174

[CGZ16] Xi Chen, Adityanand Guntuboyina, and Yuchen Zhang. On bayes risk lower bounds. *Journal of Machine Learning Research*, 17(218):1–58, 2016. 10, 88, 92

[Cha02] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002. 193

[Che09] Ke Chen. On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009. 62

[CIW24] Justin Y. Chen, Piotr Indyk, and David P. Woodruff. Space-optimal profile estimation in data streams with applications to symmetric functions. In *ITCS*, volume 287 of *LIPIcs*, pages 32:1–32:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. 205

[CJ19] Graham Cormode and Hossein Jowhari. L_p samplers and their applications: A survey. *ACM Computing Surveys (CSUR)*, 52(1):1–31, 2019. 223

[CKL13] Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *Journal of the ACM (JACM)*, 60(5):31, 2013. 31, 33, 49, 50, 51

[CKP$^+$21] Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh Saxena, Zhao Song, and Huacheng Yu. Near-optimal two-pass streaming algorithm for sampling random walks over directed graphs. *arXiv preprint arXiv:2102.11251*, 2021. 151

[CKS03] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th IEEE Annual Conference on Computational Complexity, 2003. Proceedings.*, pages 107–117. IEEE, 2003. 198, 336

[CKST17] Charles Carlson, Alexandra Kolla, Nikhil Srivastava, and Luca Trevisan. Optimal lower bounds for sketching graph cuts. *CoRR*, abs/1712.10261, 2017. 135

[CLD+20] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020. 173, 174, 175, 176, 178, 187

[CLLR17] Arkadev Chattopadhyay, Michael Langberg, Shi Li, and Atri Rudra. Tight network topology dependent bounds on rounds of communication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2524–2539. SIAM, 2017. 339

[CLM+15] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015. 30

[CMM17] Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017. 30, 258

[CMP16] Michael B Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. *arXiv preprint arXiv:1604.05448*, 2016. 423

[CMY11] Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. *ACM Trans. Algorithms*, 7(2):21:1–21:20, 2011. 297, 298

[CND+22] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022. 173

[CND+23] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. 176

[CNW15] Michael B Cohen, Jelani Nelson, and David P Woodruff. Optimal approximate matrix product in terms of stable rank. *arXiv preprint arXiv:1507.02268*, 2015. 30, 134, 135, 138, 141, 142

[Coh16] Michael B Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In

Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms, pages 278–287. SIAM, 2016. 8, 24, 29, 150

[Cor05] Graham Cormode. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005. 3

[CP11] Emmanuel J Candés and Yaniv Plan. Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *IEEE Transactions on Information Theory*, 57(4):2342–2359, 2011. 10, 81, 82, 84, 90

[CP14] Timothy M Chan and Vinayak Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Computational Geometry*, 47(2):240–247, 2014. 255

[CP15] Michael B Cohen and Richard Peng. L_p row sampling by Lewis weights. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 183–192, 2015. 7, 25, 65, 69, 70, 77, 354

[CP19] Xue Chen and Eric Price. Active regression via linear-sample sparsification. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 663–695, 2019. 134

[CPT15] Tobias Christiani, Rasmus Pagh, and Mikkel Thorup. From independence to expansion and back again. In *STOC'15—Proceedings of the 2015 ACM Symposium on Theory of Computing*, pages 813–820. ACM, New York, 2015. 203, 204, 209

[CRR14] Arkadev Chattopadhyay, Jaikumar Radhakrishnan, and Atri Rudra. Topology matters in communication. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 631–640. IEEE Computer Society, 2014. 339

[CRT06] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1207–1223, 2006. 81

[CSWZ23] Yeshwanth Cherapanamjeri, Sandeep Silwal, David P Woodruff, and Samson Zhou. Optimal algorithms for linear algebra in the current matrix multiplication time. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4026–4049. SIAM, 2023. 59

[CW09] Kenneth L Clarkson and David Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 205–214, 2009. 247, 353

[CW15] Kenneth L Clarkson and David P Woodruff. Input sparsity and hardness for robust subspace approximation. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 310–329. IEEE, 2015. 23, 65, 66, 68, 70, 71, 256, 298, 382, 385

[CW17] Kenneth L Clarkson and David Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017. 2, 6, 8, 30, 123, 150, 274, 353, 354

[CYD18] Agniva Chowdhury, Jiasen Yang, and Petros Drineas. An iterative, sketching-based framework for ridge regression. In *International conference on machine learning*, pages 989–998. PMLR, 2018. 12, 149, 150, 151, 155, 157, 158, 170

[Dao23] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. 173, 178

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019. 173

[DFE$^+$22] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. 173, 178

[Die96] Martin Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings*, volume 1046 of *Lecture Notes in Computer Science*, pages 569–580. Springer, 1996. 196, 212

[DKM06a] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating Matrix Multiplication. *SIAM J. Comput.*, 36(1):132–157, 2006. 134

[DKM06b] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36(1):158–183, 2006. 134

[DKM06c] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. *SIAM J. Comput.*, 36(1):184–206, 2006. 134

[DKS10] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse Johnson-Lindenstrauss transform. In *STOC'10—Proceedings of the 2010 ACM International Symposium on Theory of Computing*, pages 341–350. ACM, New York, 2010. 193, 207

[DKS12] Anirban Dasgupta, Ravi Kumar, and D. Sivakumar. Sparse and lopsided set disjointness via information theory. In *Approximation, randomization, and combinatorial optimization*, volume 7408 of *Lecture Notes in Comput. Sci.*, pages 517–528. Springer, Heidelberg, 2012. 247

[DMD$^+$23] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023. 174

[DMM06a] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Sampling algorithms for l_2 regression and applications. In *SODA*, pages 1127–1136, 2006. 2

[DMM06b] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-row-based methods. In *ESA*, pages 304–314, 2006. 2, 7

[DMMS11] Petros Drineas, Michael W. Mahoney, S. Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):217–249, 2011. 134

[DMMW12] Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *J. Mach. Learn. Res.*, 13:3475–3506, 2012. 33

[DP23] Amit Deshpande and Rameshwar Pratap. One-pass additive-error subset selection for ℓ_p subspace approximation and (k, p)-clustering. *Algorithmica*, pages 1–24, 2023. 256

[DPS11] Matei David, Periklis A Papakonstantinou, and Anastasios Sidiropoulos. How strong is Nisan's pseudo-random generator? *Information processing letters*, 111(16):804–808, 2011. 200, 240

[DR96] Devdatt P Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *BRICS Report Series*, 3(25), 1996. 52

[DRVW06] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(1):225–247, 2006. 2, 34, 55, 56, 62

[DTV11] Amit Deshpande, Madhur Tulsiani, and Nisheeth K Vishnoi. Algorithms and hardness for subspace approximation. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 482–496. SIAM, 2011. 253, 256

[DV07] Amit Deshpande and Kasturi Varadarajan. Sampling-based dimension reduction for subspace approximation. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 641–650, 2007. 2, 62, 256

[DWZ$^+$19] Chen Dan, Hong Wang, Hongyang Zhang, Yuchen Zhou, and Pradeep K Ravikumar. Optimal analysis of subset-selection based l_p low-rank approximation. *Advances in Neural Information Processing Systems*, 32, 2019. 257

[EEM19] Alessandro Epasto, Hossein Esfandiari, and Vahab Mirrokni. On-device algorithms for public-private data with absolute privacy. In *The World Wide Web Conference*, pages 405–416. ACM, 2019. 339, 340

[EKM$^+$24a] Hossein Esfandiari, Praneeth Kacham, Vahab Mirrokni, David Woodruff, and Peilin

Zhong. Optimal communication bounds for classic functions in the coordinator model and beyond. *arXiv:2403.20307*, 2024. STOC. 18, 20

[EKM+24b] Hossein Esfandiari, Praneeth Kacham, Vahab Mirrokni, David Woodruff, and Peilin Zhong. Space-efficient algorithms for high-dimensional geometric streaming for almost low rank data. *Proceedings of Machine Learning*, 2024. ICML. 17, 20

[FIS05] Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 142–149, 2005. 193

[FKV04] Alan M. Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004. 2

[FKW21] Zhili Feng, Praneeth Kacham, and David Woodruff. Dimensionality reduction for the sum-of-distances metric. In *International conference on machine learning*, pages 3220–3229. PMLR, 2021. 20

[FL11] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 569578, New York, NY, USA, 2011. Association for Computing Machinery. 7, 62, 78

[FMSW10] Dan Feldman, Morteza Monemizadeh, Christian Sohler, and David P Woodruff. Coresets and sketches for high dimensional subspace approximation problems. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 630–649. Society for Industrial and Applied Mathematics, 2010. 62, 256

[FNM07] Tom Fawcett and Alexandru Niculescu-Mizil. Pav and the roc convex hull. *Machine Learning*, 68:97–106, 2007. 257

[FR13] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser/Springer, New York, 2013. 117

[FS05] Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 209–217. ACM, 2005. 62

[FS08] Gereon Frahling and Christian Sohler. A fast k-means implementation using coresets. *International Journal of Computational Geometry & Applications*, 18(06):605–625, 2008. 62

[FS12] Dan Feldman and Leonard J Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1343–1354. Society for Industrial and Applied Mathematics, 2012. 62

[FSS13] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the* *twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1434–1453. Society for Industrial and Applied Mathematics, 2013. 62

[FSS20] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM Journal on Computing*, 49(3):601–657, 2020. 340

[FT07] Shmuel Friedland and Anatoli Torokhti. Generalized rank-constrained matrix approximations. *SIAM Journal on Matrix Analysis and Applications*, 29(2):656–659, 2007. 119

[Gan15] Sumit Ganguly. Taylor polynomial estimator for estimating frequency moments. In *Automata, languages, and programming. Part I*, volume 9134 of *Lecture Notes in Comput. Sci.*, pages 542–553. Springer, Heidelberg, 2015. 199

[GD23] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 176

[GDVAR20] Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. Wiki-40b: Multilingual language model dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2440–2452, 2020. 189

[Gha19] Mohsen Ghaffari. Distributed maximal independent set using small messages. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 805–820. SIAM, 2019. 339

[GKM18] Parikshit Gopalan, Daniel M. Kane, and Raghu Meka. Pseudorandomness via the discrete fourier transform. *SIAM Journal on Computing*, 47(6):2451–2487, 2018. 201

[GKMS03] Anna C. Gilbert, Yannis Kotidis, S Muthukrishnan, and Martin J Strauss. One-pass wavelet decompositions of data streams. *IEEE Transactions on knowledge and data engineering*, 15(3):541–554, 2003. 193

[GKS17] Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed MST and routing in almost mixing time. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 131–140, 2017. 339

[GL18] Mohsen Ghaffari and Jason Li. Improved distributed algorithms for exact shortest paths. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 431–444, 2018. 339

[GLM$^+$21] Rachel Grotheer, Shuang Li, Anna Ma, Deanna Needell, and Jing Qin. Iterative hard thresholding for low cp-rank tensor models. *Linear and Multilinear Algebra*, pages 1–17, 2021. 85

[GLO81] Gene H Golub, Franklin T Luk, and Michael L Overton. A block lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software (TOMS)*, 7(2):149–169, 1981. 114

[GLPS12] Anna C. Gilbert, Yi Li, Ely Porat, and Martin J. Strauss. Approximate sparse recovery: optimizing time and measurements. *SIAM J. Comput.*, 41(2):436–453, 2012. 81

[GLPW16] Mina Ghashami, Edo Liberty, Jeff M Phillips, and David P Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792, 2016. 274

[GM09] Sudipto Guha and Andrew McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009. 275

[GMV05] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. *arXiv preprint cs/0508122*, 2005. 275

[Gro09] André Gronemeier. Asymptotically optimal lower bounds on the NIH-Multi-Party information complexity of the and-function and disjointness. In *STACS*, volume 3 of *LIPIcs*, pages 505–516. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. 336

[Gu15] Ming Gu. Subspace iteration randomization and singular value problems. *SIAM Journal on Scientific Computing*, 37(3):A1139–A1173, 2015. 83, 109, 113, 273, 276, 401

[GW18] Sumit Ganguly and David P. Woodruff. High probability frequency moment sketches. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 58, 15. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018. 199

[HAS20] Insu Han, Haim Avron, and Jinwoo Shin. Polynomial tensor sketch for element-wise function of low-rank matrix. In *International Conference on Machine Learning*, pages 3984–3993. PMLR, 2020. 176

[HDLL22] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International Conference on Machine Learning*, pages 9099–9117. PMLR, 2022. 175, 176, 177

[HFB15] Wooseok Ha and Rina Foygel Barber. Robust PCA with compressed data. *Advances in Neural Information Processing Systems*, 28, 2015. 90

[HIKV19] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2019. 177, 184

[HJK$^+$23] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. *arXiv preprint arXiv:2310.05869*, 2023. 173, 174

[HL18] Bernhard Haeupler and Jason Li. Faster Distributed Shortest Path Approximations via Shortcuts. In *32nd International Symposium on Distributed Computing (DISC 2018)*, volume

121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 339

[HM13] Moritz Hardt and Ankur Moitra. Algorithms and hardness for robust subspace recovery. In *Conference on Learning Theory*, pages 354–375. PMLR, 2013. 257

[HMT11] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011. 120

[HNO08] Nicholas JA Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 489–498. IEEE, 2008. 205

[HNWW21] De Huang, Jonathan Niles-Weed, and Rachel Ward. Streaming k-PCA: Efficient guarantees for ojas algorithm, beyond rank-one updates. In *Conference on Learning Theory*, pages 2463–2498. PMLR, 2021. 274

[HP14] Moritz Hardt and Eric Price. The noisy power method: A meta algorithm with applications. *Advances in neural information processing systems*, 27, 2014. 123, 274

[HPK07] Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007. 62

[HPM04] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM, 2004. 62

[HR13] Moritz Hardt and Aaron Roth. Beyond worst-case analysis in private singular vector computation. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 331340, New York, NY, USA, 2013. Association for Computing Machinery. 123

[HV20] Lingxiao Huang and Nisheeth K. Vishnoi. Coresets for clustering in Euclidean spaces: Importance sampling is nearly optimal. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 14161429, New York, NY, USA, 2020. Association for Computing Machinery. 63, 79

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006. 3, 16, 193, 194, 199

[Ind07] Piotr Indyk. Uncertainty principles, extractors, and explicit embeddings of ℓ_2 into ℓ_1. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 615–620, 2007. 8, 29, 32, 34, 35

[IPW11] Piotr Indyk, Eric Price, and David P. Woodruff. On the power of adaptivity in sparse recovery. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*. IEEE

Computer Society, 2011. 82

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. 175

[IW05] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 202–208. ACM, New York, 2005. 3, 15, 198, 199

[JAX23] JAX authors. Implementation of FlashAttention in Pallas. https://github.com/google/jax/blob/main/jax/experimental/pallas/ops/attention.py, 2023. 187

[Jay09] T. S. Jayram. Hellinger strikes back: a note on the multi-party information complexity of AND. In *Approximation, randomization, and combinatorial optimization*, volume 5687 of *Lecture Notes in Comput. Sci.*, pages 562–573. Springer, Berlin, 2009. 198

[JJK$^+$16] Prateek Jain, Chi Jin, Sham M Kakade, Praneeth Netrapalli, and Aaron Sidford. Streaming pca: Matching matrix bernstein and near-optimal finite sample guarantees for ojas algorithm. In *Conference on learning theory*, pages 1147–1164. PMLR, 2016. 274

[JLSW20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 944–953, 2020. 120

[JM17] Tiefeng Jiang and Yutao Ma. Distances between random orthogonal matrices and independent normals. *arXiv preprint arXiv:1704.05205*, 2017. 153

[JST11] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for ℓ_p samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58, 2011. 6, 193, 225, 418

[JW18] Rajesh Jayaram and David P. Woodruff. Perfect L_p sampling in a data stream. In *Symposium on Foundations of Computer Science (FOCS)*, pages 544–555, 2018. 201, 223, 237, 238

[JW21] Rajesh Jayaram and David Woodruff. Perfect l_p sampling in a data stream. *SIAM Journal on Computing*, 50(2):382–439, 2021. 6

[KBW$^+$19] Dmitry Kobak, Yves Bernaerts, Marissa A Weis, Federico Scala, Andreas Tolias, and Philipp Berens. Sparse reduced-rank regression for exploratory visualization of multimodal data sets. *bioRxiv*, page 302208, 2019. 119

[KKL20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020. 174

[KL15] Zohar Karnin and Edo Liberty. Online pca with spectral bounds. In *Conference on Learning Theory*, pages 1129–1140, 2015. 120

[KMZ23] Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655*, 2023. 20

[KN21] Christian Konrad and Kheeran K Naidu. On two-pass streaming algorithms for maximum bipartite matching. *arXiv preprint arXiv:2107.07841*, 2021. 151

[KNPW11] Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. Fast moment estimation in data streams in optimal space. In *STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 745–754. ACM, New York, 2011. xiii, 6, 15, 16, 197, 199, 200, 204, 205, 207, 224, 225, 226, 227, 228, 252

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1161–1178. SIAM, Philadelphia, PA, 2010. 3, 15, 16, 193, 199, 418

[KP98] Shay Kutten and David Peleg. Fast distributed construction of small k-dominating sets and applications. *Journal of Algorithms*, 28(1):40–66, 1998. 339

[KP19] Akshay Kamath and Eric Price. Adaptive sparse recovery with limited adaptivity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2729–2744. SIAM, Philadelphia, PA, 2019. 82

[KPTW23] Praneeth Kacham, Rasmus Pagh, Mikkel Thorup, and David P. Woodruff. Pseudorandom hashing for space-bounded computation with applications in streaming. *arXiv preprint arXiv:2304.06853*, 2023. FOCS. 16, 20

[KPZ$^+$21] Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A Smith. Finetuning pretrained transformers into rnns. *arXiv preprint arXiv:2103.13076*, 2021. 175

[KR14] Michael Kerber and Sharath Raghvendra. Approximation and streaming algorithms for projective clustering via random projections. *arXiv preprint arXiv:1407.2063*, 2014. 256

[KS24] Syamantak Kumar and Purnamrita Sarkar. Streaming PCA for Markovian data. *Advances in Neural Information Processing Systems*, 36, 2024. 274

[KVPF20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020. 173, 174, 175, 176

[KVW14] Ravi Kannan, Santosh Vempala, and David Woodruff. Principal component analysis and higher correlations for distributed data. In *Conference on Learning Theory*, pages 1040–1057. PMLR, 2014. xv, 4, 6, 298, 300, 303, 330, 338, 340, 425

[KVW18] Ravi Kannan, Santosh Vempala, and David Woodruff. Personal communication, 2018. 298

[KW20] Praneeth Kacham and David Woodruff. Optimal deterministic coresets for ridge regression. In *International Conference on Artificial Intelligence and Statistics*, pages 4141–4150. PMLR, 2020. 13, 20

[KW21] Praneeth Kacham and David Woodruff. Reduced-rank regression with operator norm error. In *Conference on Learning Theory*, pages 2679–2716. PMLR, 2021. 20

[KW22] Praneeth Kacham and David Woodruff. Sketching algorithms and lower bounds for ridge regression. In *International Conference on Machine Learning*, pages 10539–10556. PMLR, 2022. 12, 20

[KW23] Praneeth Kacham and David Woodruff. Lower bounds on adaptive sensing for matrix recovery. *arXiv:2311.17281*, 2023. NeurIPS. 10, 20

[KW24] Praneeth Kacham and David P. Woodruff. Faster algorithms for Schatten-p low rank approximation. Under review, 2024. 132

[LA99] Pierre Legendre and Marti J Anderson. Distance-based redundancy analysis: testing multispecies responses in multifactorial ecological experiments. *Ecological monographs*, 69(1):1–24, 1999. 119

[LBKW14] Yingyu Liang, Maria-Florina F Balcan, Vandana Kanchanapally, and David Woodruff. Improved distributed principal component analysis. *Advances in neural information processing systems*, 27, 2014. 68, 79

[Lew78] D. Lewis. Finite dimensional subspaces of L_p. *Studia Mathematica*, 63(2):207–212, 1978. 25, 354

[Li08] Ping Li. Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–19. ACM, New York, 2008. 3, 199, 204, 225

[LM00] Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000. 46, 94, 104, 106

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L. Nguyen, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *57th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2016*, pages 61–70. IEEE Computer Soc., Los Alamitos, CA, 2016. 418

[LNW14] Yi Li, Huy L. Nguyen, and David P. Woodruff. On sketching matrix norms and the top singular vector. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1562–1581. SIAM, 2014. 111

[Low12] George Lowther. Anti-concentration of Gaussian Quadratic form. MathOverflow.net,

2012. 163

[LS10] Michael Langberg and Leonard J Schulman. Universal ε-approximators for integrals. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 598–607. SIAM, 2010. 62

[LS18] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. *SIAM J. Comput.*, 47(6):2315–2336, 2018. 134

[Lub86] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986. 339

[LW16] Yi Li and David P. Woodruff. Tight bounds for sketching the operator norm, Schatten norms, and subspace embeddings. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, volume 60 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 39, 11. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016. 90

[LW21] Yi Li and David P. Woodruff. The product of Gaussian matrices is close to Gaussian. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, 2021. 153, 165

[Mah07] PJ Maher. Some norm inequalities concerning generalized inverses, 2. *Linear algebra and its applications*, 420(2-3):517–525, 2007. 121

[Mah11] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011. 134

[Mat13] Jirı Matoušek. Lecture notes on metric embeddings. Technical report, Technical report, ETH Zürich, 2013. 385

[MI10] Malik Magdon-Ismail. Row sampling for matrix algorithms via a non-commutative bernstein bound. *arXiv preprint arXiv:1008.0587*, 2010. 276, 277

[MJF19] Alaa Maalouf, Ibrahim Jubran, and Dan Feldman. Fast and accurate least-mean-squares solvers. *CoRR*, abs/1906.04705, 2019. 134, 139

[MM13] Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 91–100, 2013. 30, 123, 150

[MM15] Cameron Musco and Christopher Musco. Randomized Block Krylov methods for stronger and faster approximate singular value decomposition. *Advances in neural information processing systems*, 28, 2015. 11, 113, 114, 115, 118, 120, 121, 122, 124, 125, 126, 127, 128, 129, 132, 273, 393, 398

[MMO22] Yury Makarychev, Naren Sarayu Manoj, and Max Ovsiankin. Streaming algorithms for ellipsoidal approximation of convex polytopes. In *Conference on Learning Theory*, pages

3070–3093. PMLR, 2022. 261

[MMR19] Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1027–1038. ACM, 2019. 63

[MMS18] Cameron Musco, Christopher Musco, and Aaron Sidford. Stability of the Lanczos method for matrix function approximation. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, page 16051624, USA, 2018. Society for Industrial and Applied Mathematics. 123, 132, 273

[MMWY22] Cameron Musco, Christopher Musco, David Woodruff, and Taisuke Yasuda. Active linear regression for ℓ_p norms and beyond. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science—FOCS 2022*, pages 744–753. IEEE Computer Soc., Los Alamitos, CA, 2022. 24, 346

[Mor78] Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978. 3

[MP80] J Ian Munro and Mike S Paterson. Selection and sorting with limited storage. *Theoretical computer science*, 12(3):315–323, 1980. 275

[MP14] Gregory T. Minton and Eric Price. Improved concentration bounds for Count-Sketch. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 669–686. ACM, New York, 2014. 16, 197, 198, 200, 207, 208, 236, 237, 238, 240

[MSW19] Michela Meister, Tamas Sarlos, and David Woodruff. Tight dimensionality reduction for sketching low degree polynomial kernels. *Advances in Neural Information Processing Systems*, 32, 2019. 176

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 158

[MW10] Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error L_p-sampling with applications. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1143–1160. SIAM, Philadelphia, PA, 2010. 199

[Nam15] Srinivas Nambirajan. *Topics in Matrix Approximation*. PhD thesis, Rensselaer Polytechnic Institute, USA, 2015. 126

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991. 336

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. 5, 16, 193, 196, 202, 210, 213, 214, 216, 237, 238, 330, 417, 426

[NN13] Jelani Nelson and Huy L. Nguyên. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of*

Computer Science, pages 117–126, 2013. 2, 8, 24, 29, 30, 123, 150, 156, 157

[NN14] Jelani Nelson and Huy L Nguyen. Lower bounds for oblivious subspace embeddings. In *International Colloquium on Automata, Languages, and Programming*, pages 883–894. Springer, 2014. 23, 152

[NNW14] Jelani Nelson, Huy L Nguyen, and David Woodruff. On deterministic sketching and streaming for sparse recovery and norm estimation. *Linear Algebra and its Applications*, 441:152–167, 2014. 301

[NSWZ18] Vasileios Nakos, Xiaofei Shi, David P. Woodruff, and Hongyang Zhang. Improved algorithms for adaptive compressed sensing. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 90, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018. 82

[NW10] Jelani Nelson and David P Woodruff. Fast Manhattan sketches in data streams. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 99–110, 2010. 228

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. System Sci.*, 52(1):43–52, 1996. 15, 198, 217

[Oja82] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15:267–273, 1982. 274

[Ope23] OpenAI. Gpt-4 technical report, 2023. 173

[Pel00] David Peleg. *Distributed computing: a locality-sensitive approach.* SIAM, 2000. 4, 339

[PF01] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine learning*, 42:203–231, 2001. 257

[PHC07] Moon-Hee Park, Jin-Hyuk Hong, and Sung-Bae Cho. Location-based recommendation system using Bayesian users preference model in mobile devices. In *International conference on ubiquitous intelligence and computing*, pages 1130–1139. Springer, 2007. 340

[PHK$^+$21] Younghyun Park, Dong-Jun Han, Do-Yeon Kim, Jun Seo, and Jaekyun Moon. Few-round learning for federated learning. *Advances in Neural Information Processing Systems*, 34, 2021. 151

[PP08] Anna Pagh and Rasmus Pagh. Uniform hashing in constant time and optimal space. *SIAM Journal on Computing*, 38(1):85–96, 2008. 204, 209

[PPY$^+$21] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021. 174, 175

[Pri23] Eric Price. Spectral guarantees for adversarial streaming PCA. https://www.cs.utexas.edu/~ecprice/papers/streaming-adversarial-pca.pdf, 2023. 18, 274, 275, 276, 287, 288

[PT22] Rasmus Pagh and Mikkel Thorup. Improved utility analysis of private CountSketch. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 197, 207, 241

[PVZ16] Jeff M. Phillips, Elad Verbin, and Qin Zhang. Lower bounds for number-in-hand multi-party communication complexity, made easy. *SIAM J. Comput.*, 45(1):174–196, 2016. 297

[PW11] Eric Price and David P. Woodruff. $(1+\varepsilon)$-approximate sparse recovery. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science—FOCS 2011*, pages 295–304. IEEE Computer Soc., Los Alamitos, CA, 2011. 81

[PW12] Eric Price and David P. Woodruff. Lower bounds for adaptive sparse recovery. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 652–663. SIAM, Philadelphia, PA, 2012. 81, 82, 90

[RPJ$^+$19] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. 189

[RSS17] Holger Rauhut, Reinhold Schneider, and Željka Stojanac. Low rank tensor recovery via iterative hard thresholding. *Linear Algebra Appl.*, 523:220–262, 2017. 85

[RSVG21] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. 174

[RV09] Mark Rudelson and Roman Vershynin. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 62(12):1707–1739, 2009. 94, 104, 106, 395

[RV10] Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: Extreme singular values. *Proceedings of the International Congress of Mathematicians 2010, ICM 2010*, 03 2010. 394, 395

[RWC$^+$19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 176, 187

[RWZ20] Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu. Vector-matrix-vector queries for solving linear algebra, statistics, and graph problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020*, volume 176 of *LIPIcs*, pages 26:1–26:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. 85

[Sai19] Arvind K Saibaba. Randomized subspace iteration: Analysis of canonical angles and unitarily invariant norms. *SIAM Journal on Matrix Analysis and Applications*, 40(1):23–48, 2019. 112

[Sar06] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*. IEEE, 2006. 2, 6, 8, 138

[SC17] Yiyuan She and Kun Chen. Robust reduced-rank regression. *Biometrika*, 104(3):633–647, 2017. 119

[SEAR18] Max Simchowitz, Ahmed El Alaoui, and Benjamin Recht. Tight query complexity lower bounds for pca via finite sample deformed wigner law. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1249–1259, 2018. 91, 109

[SKT14] Arthur Szlam, Yuval Kluger, and Mark Tygert. An implementation of a randomized algorithm for principal component analysis. *arXiv preprint arXiv:1412.3510*, 2014. 120

[SR12] Kin Cheong Sou and Anders Rantzer. On generalized matrix approximation problem in the spectral norm. *Linear Algebra and its Applications*, 436(7):2331–2341, 2012. 11, 121, 122, 126, 127

[SV07] Nariankadu D Shyamalkumar and Kasturi Varadarajan. Efficient subspace approximation algorithms. In *SODA*, volume 7, pages 532–540, 2007. 62

[SV14] Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trendső in Theoretical Computer Science*, 9(2):125–210, 2014. 123, 405

[SW18] Christian Sohler and David P Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 802–813. IEEE, 2018. 9, 62, 63, 65, 66, 67, 68, 76, 79

[SWYZ21a] Zhao Song, David Woodruff, Zheng Yu, and Lichen Zhang. Fast sketching of polynomial kernels of polynomial degree. In *International Conference on Machine Learning*, pages 9812–9823. PMLR, 2021. 176

[SWYZ21b] Xiaoming Sun, David P. Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. *ACM Trans. Algorithms*, 17(4):31:1–31:19, 2021. 85

[SWZ19] Zhao Song, David Woodruff, and Peilin Zhong. Towards a zero-one law for column subset selection. *Advances in Neural Information Processing Systems*, 32, 2019. 257

[SYY21] Zhiqing Sun, Yiming Yang, and Shinjae Yoo. Sparse attention with learning to hash. In *International Conference on Learning Representations*, 2021. 174

[TB97] Lloyd N. Trefethen and David Bau. *Numerical linear algebra*. SIAM, 1997. 2

[TBY$^+$19] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: a unified understanding of transformer's attention via the lens of kernel. *arXiv preprint arXiv:1908.11775*, 2019. 174, 175

[TDBM22] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2022. 173

[Tro11] Joel A Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011. 2, 8

[Tro15] Joel A Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015. 277

[TV23] Jared Tanner and Simon Vary. Compressed sensing of low-rank plus sparse matrices. *Applied and Computational Harmonic Analysis*, 2023. 90

[TWZ$^+$22] Murad Tukan, Xuan Wu, Samson Zhou, Vladimir Braverman, and Dan Feldman. New coresets for projective clustering and applications. In *International Conference on Artificial Intelligence and Statistics*, pages 5391–5415. PMLR, 2022. 255

[Upa16] Jalaj Upadhyay. Fast and space-optimal low-rank factorization in the streaming model with application in differential privacy. *arXiv preprint arXiv:1604.01429*, 2016. 274

[Ver20] Roman Vershynin. Concentration inequalities for random tensors. *Bernoulli*, 26(4):3139 – 3162, 2020. 85, 88, 96

[VR13] Raja Velu and Gregory C Reinsel. *Multivariate reduced-rank regression: theory and applications*, volume 136. Springer Science & Business Media, 2013. 119

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 13, 173, 179

[VT07] Van H. Vu and Terence Tao. The condition number of a randomly perturbed matrix. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 248255, New York, NY, USA, 2007. Association for Computing Machinery. 415

[VVYZ07] Kasturi Varadarajan, S. Venkatesh, Yinyu Ye, and Jiawei Zhang. Approximating the radii of point sets. *SIAM J. Comput.*, 36(6):1764–1776, 2007. 253, 255

[VWW19] Santosh S. Vempala, Ruosong Wang, and David P. Woodruff. The communication complexity of optimization. *CoRR*, abs/1906.05832, 2019. 135

[VX12] Kasturi Varadarajan and Xin Xiao. On the sensitivity of shape fitting problems. *arXiv preprint arXiv:1209.4893*, 2012. 7, 62

[Waj17] David Wajc. Negative association - definition, properties, and applications, 2017. 52

[WLK$^+$20] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 173, 174

[Woo14] David Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014. 66, 68, 74, 115, 121, 125, 134, 180, 182, 384

[WW] Gloria Wang and Zirui Wang. Physics Multiple Choice. 189

[WW19] Ruosong Wang and David P. Woodruff. Tight bounds for lp oblivious subspace embeddings. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, page 18251843, USA, 2019. Society for Industrial and Applied Mathematics. 65, 66

[WWZ14] Karl Wimmer, Yi Wu, and Peng Zhang. Optimal query complexity for estimating the trace of a matrix. In *International Colloquium on Automata, Languages, and Programming*, pages 1051–1062. Springer, 2014. 85

[WX97] Bo-Ying Wang and Bo-Yan Xi. Some inequalities for singular values of matrix products. *Linear algebra and its applications*, 264:109–115, 1997. 276, 282

[WXXZ24] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3792–3835. SIAM, 2024. 29

[WY22] David P. Woodruff and Taisuke Yasuda. High-dimensional geometric streaming in polynomial space. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science—FOCS 2022*, pages 732–743. IEEE Computer Soc., Los Alamitos, CA, 2022. 255, 256, 264, 271, 272

[WY23] David P Woodruff and Taisuke Yasuda. New subset selection algorithms for low rank approximation: Offline and online. *arXiv preprint arXiv:2304.09217*, 2023. 256

[WZ12] David Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *STOC'12—Proceedings of the 2012 ACM Symposium on Theory of Computing*. ACM, New York, 2012. 19, 298, 299, 300, 331, 332, 333, 339

[WZ13] David Woodruff and Qin Zhang. Subspace embeddings and ℓ_p-regression using exponential random variables. In *Conference on Learning Theory*, pages 546–567. PMLR, 2013. 340

[WZ17] David Woodruff and Qin Zhang. When distributed computation is communication expensive. *Distributed Comput.*, 30(5):309–323, 2017. 297

[XTC$^+$23] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023. 174

[YWS$^+$23] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023. 175, 176

[ZGD$^+$20] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020. 174

[ZHB$^+$19] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019. 189

[ZJD15] Kai Zhong, Prateek Jain, and Inderjit S Dhillon. Efficient matrix sensing using rank-1 gaussian measurements. In *International conference on algorithmic learning theory*, pages 3–18. Springer, 2015. 81

[ZKXL19] Wei Zhang, Taejoon Kim, Guojun Xiong, and Shu-Hung Leung. Leveraging subspace information for low-rank matrix reconstruction. *Signal processing*, 163:123–131, 2019. 82

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures Algorithms*, 11(4):345–367, 1997. 35

Appendix A

Deferred Proofs from Chapter 4

A.1 Lopsided Embeddings and Gaussian Matrices

Recall $\|\cdot\|_h$ is defined as $\|A\|_h = \sum_j \|A_{*j}\|_2$. Note that $\|A\|_h = \|A^{\mathrm{T}}\|_{1,2}$ for all matrices A. The following lemma shows that lopsided-ε embeddings for certain matrices w.r.t. the norm $\|\cdot\|_h$ imply a dimension reduction for $\|\cdot\|_{1,2}$ subspace approximation.

Lemma A.1.1. *Given a matrix $A \in \mathbb{R}^{n\times d}$ and a parameter $k \in \mathbb{Z}_{>0}$, let $U_k \in \mathbb{R}^{n\times k}$ and $V_k^{\mathrm{T}} \in \mathbb{R}^{k\times d}$ be matrices such that*

$$\|U_k V_k^{\mathrm{T}} - A\|_{1,2} = \min_{\text{rank-}k\ X} \|A(I - X)\|_{1,2}.$$

If S is a lopsided ε-embedding for (V_k, A^{T}) with respect to the norm $\|\cdot\|_h$, then

$$\min_{\text{rank-}k\ X} \|AS^{\mathrm{T}}X - A\|_{1,2} \le (1 + O(\varepsilon)) \min_{\text{rank-}k\ X} \|A(I - X)\|_{1,2}.$$

Proof. Note that $\|V_k U_k^{\mathrm{T}} - A^{\mathrm{T}}\|_h = \min_Y \|V_k Y^{\mathrm{T}} - A^{\mathrm{T}}\|_h$. By definition of a lopsided embedding, we have the following for any matrix Y:

$$\|Y V_k^{\mathrm{T}} S^{\mathrm{T}} - AS^{\mathrm{T}}\|_{1,2} = \|SV_k Y^{\mathrm{T}} - SA^{\mathrm{T}}\|_h \ge (1-\varepsilon)\|V_k Y^{\mathrm{T}} - A^{\mathrm{T}}\|_h = (1-\varepsilon)\|Y V_k^{\mathrm{T}} - A\|_{1,2}$$

and also that

$$\|U_k V_k^{\mathrm{T}} S^{\mathrm{T}} - AS^{\mathrm{T}}\|_{1,2} = \|SV_k U_k^{\mathrm{T}} - SA^{\mathrm{T}}\|_h \le (1+\varepsilon)\|V_k U_k^{\mathrm{T}} - A^{\mathrm{T}}\|_h = (1+\varepsilon)\|U_k V_k^{\mathrm{T}} - A\|_{1,2}.$$

Using these guarantees we now show that the column span of the matrix AS^{T} contains a good solution to the subspace approximation problem. First consider the minimization problem

$$\min_Y \|Y V_k^{\mathrm{T}} - A\|_{1,2}.$$

Clearly, U_k is the optimal solution to the problem. Now consider the optimal solution $\widetilde{Y}$ to the sketched version of the above problem

$$\widetilde{Y} = \arg\min_{Y} \|YV_k^{\mathrm{T}}S^{\mathrm{T}} - AS^{\mathrm{T}}\|_{1,2}.$$

We can see that $\widetilde{Y} = (AS^{\mathrm{T}})(V_k^{\mathrm{T}}S^{\mathrm{T}})^{+}$. Now

$$\|\widetilde{Y}V_k^{\mathrm{T}} - A\|_{1,2} \le \frac{1}{1-\varepsilon}\|\widetilde{Y}V_k^{\mathrm{T}}S^{\mathrm{T}} - AS^{\mathrm{T}}\|_{1,2} \le \frac{1}{1-\varepsilon}\|U_K V_k^{\mathrm{T}}S^{\mathrm{T}} - AS^{\mathrm{T}}\| \le \frac{1+\varepsilon}{1-\varepsilon}\|U_k V_k^{\mathrm{T}} - A\|_{1,2}.$$

Therefore,

$$\min_{\text{rank-k } X} \|AS^{\mathrm{T}}X - A\|_{1,2} \le \|AS^{\mathrm{T}}(V_k^{\mathrm{T}}S^{\mathrm{T}})^{+}(V_k^{\mathrm{T}}) - A\|_{1,2} \le \frac{1+\varepsilon}{1-\varepsilon}\|U_k V_k^{\mathrm{T}} - A\|_{1,2} \le (1+3\varepsilon) \min_{\text{rank-}k\ X} \|A(I-X)\|_{1,2}.$$

Thus, if the number of rows of S is less than d, we obtain a dimension reduction for $\|\cdot\|_{1,2}$ subspace approximation. □

Clarkson and Woodruff [CW15] give the following sufficient conditions for a distribution of matrices to be an ε-lopsided embedding for (A, B) with respect to $\|\cdot\|_h$. For the sake of completeness we reproduce their proof here.

Lemma A.1.2 (Sufficient Conditions). *Given matrices (A, B), let $\mathbf{S}$ be a matrix drawn from a distribution such that*

1. *the matrix $\mathbf{S}$ is a subspace ε-contraction for A with respect to $\|\cdot\|_2$, i.e., simultaneously for all vectors x*

$$\|\mathbf{S}Ax\|_2 \ge (1-\varepsilon)\|Ax\|_2$$

   with probability $1 - \delta/3$,
2. *for all $i \in [d']$, with probability at least $1-\delta\varepsilon^2/3$ the matrix $\mathbf{S}$ is a subspace ε^2-contraction for $[A\, B_{*i}]$ with respect to $\|\cdot\|_2$, i.e., for all vectors x,*

$$\|\mathbf{S}Ax - \mathbf{S}B_{*i}\|_2 \ge (1-\varepsilon^2)\|Ax - B_{*i}\|_2,$$

   and
3. *the matrix $\mathbf{S}$ is an ε^2-dilation for B^* with respect to $\|\cdot\|_h$, i.e., $\|\mathbf{S}B^*\|_h \le (1+\varepsilon^2)\|B^*\|_h$ with probability $\ge 1 - \delta/3$.*

In the Condition 3 above, $B^* = AX^* - B$ where $X^* = \arg\min_X \|AX - B\|_h$. With failure probability at most δ, the matrix $\mathbf{S}$ is an affine 6ε-contraction for (A, B) with respect to $\|\cdot\|_h$, i.e., for all matrices X,

$$\|\mathbf{S}(AX - B)\|_h \ge (1-6\varepsilon)\|AX - B\|_h$$

and therefore a lopsided 6ε-embedding for (A, B) with respect to $\|\cdot\|_h$.

Importantly, note that Condition 2 in the lemma is about the probability of $\boldsymbol{S}$ being a subspace contraction for $[A\, B_{*i}]$ separately for each i and *not* the probability of $\boldsymbol{S}$ being *simultaneously* a subspace contraction for $[A\, B_{*i}]$ for all $i \in [d']$.

Proof. Condition on the event that 1 and 3 hold. For $i \in [d']$, let $\boldsymbol{Z}_i$ be an indicator random variable where $\boldsymbol{Z}_i = 0$ if the matrix $\boldsymbol{S}$ is a subspace ε^2-contraction for $[A\, B_{*i}]$ and $\boldsymbol{Z}_i = 1$ otherwise. From the properties of $\boldsymbol{S}$, we have that $\mathbf{Pr}[\boldsymbol{Z}_i = 1] \le \delta\varepsilon^2/3$ for all i. If $\boldsymbol{Z}_i = 1$, we call i *bad* and if $\boldsymbol{Z}_i = 0$, we call i *good*.

Consider an arbitrary matrix X. Say a *bad i* is *large* if $\|(AX - B)_{*i}\|_2 \ge (1/\varepsilon)(\|B^*_{*i}\|_2 + \|\boldsymbol{S}B^*_{*i}\|_2)$, otherwise a *bad i* is *small*. We have

$$\sum_{\textit{small } i} \|(AX - B)_{*i}\|_2 \le (1/\varepsilon) \sum_{\textit{small } i} \|B^*_{*i}\|_2 + \|\boldsymbol{S}B^*_{*i}\|_2 \le (1/\varepsilon) \sum_{\textit{bad } i} \|B^*_{*i}\|_2 + \|\boldsymbol{S}B^*_{*i}\|_2. \tag{A.1}$$

Using condition 2, we obtain that $\mathbf{E}[\sum_{\textit{bad } i} \|B^*_{*i}\|_2] \le (\delta\varepsilon^2/3) \sum_i \|B^*_{*i}\|_2 \le (\delta\varepsilon^2/3)\Delta^*$. By a Markov bound, we have that with probability $\ge 1 - \delta/3$, $\sum_{\textit{bad } i} \|B^*_{*i}\| \le \varepsilon^2\Delta^*$. Assume that this event holds. Similarly,

$$\begin{aligned}
\sum_{\textit{bad } i} \|\boldsymbol{S}B^*_{*i}\|_2 &= \|\boldsymbol{S}B^*\|_h - \sum_{\textit{good } i} \|\boldsymbol{S}B^*_{*i}\|_2 \\
&\le (1+\varepsilon^2)\Delta^* - (1-\varepsilon^2) \sum_{\textit{good } i} \|B^*_{*i}\|_2 \\
&\le (1+\varepsilon^2)\Delta^* - (1-\varepsilon^2)(\Delta^* - \varepsilon^2\Delta^*) \\
&\le 3\varepsilon^2\Delta^*.
\end{aligned}$$

Thus, we can bound the RHS of (A.1) and obtain

$$\sum_{\textit{small } i} \|(AX - B)_{*i}\|_2 \le (1/\varepsilon)(\varepsilon^2\Delta^* + 3\varepsilon^2\Delta^*) \le 4\varepsilon\Delta^*.$$

Now we lower bound $\sum_{\textit{bad } i} \|\boldsymbol{S}(AX - B)_{*i}\|_2$.

$$\begin{aligned}
\sum_{\textit{bad } i} \|\boldsymbol{S}(AX - B)_{*i}\|_2 &\ge \sum_{\textit{large } i} \|\boldsymbol{S}(AX - B)_{*i}\|_2 \\
&\ge \sum_{\textit{large } i} \|\boldsymbol{S}(AX - AX^*)_{*i}\|_2 - \|\boldsymbol{S}B^*_{*i}\|_2 \\
&\ge \sum_{\textit{large } i} (1-\varepsilon)\|(AX - AX^*)_{*i}\|_2 - \|\boldsymbol{S}B^*_{*i}\|_2 \\
&\ge \sum_{\textit{large } i} (1-\varepsilon)\|(AX - B)_{*i}\|_2 - (1-\varepsilon)\|B^*_{*i}\|_2 - \|\boldsymbol{S}B^*_{*i}\|_2
\end{aligned}$$

$$\geq \sum_{\text{large } i} (1-\varepsilon)\|(AX-B)_{*i}\|_2 - \varepsilon\|(AX-B)_{*i}\|_2$$
$$\geq (1-2\varepsilon) \sum_{\text{large } i} \|(AX-B)_{*i}\|_2.$$

In the above, we repeatedly used the triangle inequality for the $\|\cdot\|_2$ norm, and that $\boldsymbol{S}$ is a subspace ε-embedding for matrix A and for large i, we upper bound $(1-\varepsilon)\|B^*_{*i}\|_2 + \|\boldsymbol{S}B^*_{*i}\|_2$ by $\varepsilon\|(AX-B)_{*i}\|_2$. We can finally lower bound $\|\boldsymbol{S}(AX-B)\|_h$.

$$\begin{aligned}
\|\boldsymbol{S}(AX-B)\|_h &= \sum_{\text{good } i} \|\boldsymbol{S}(AX-B)_{*i}\|_2 + \sum_{\text{bad } i} \|\boldsymbol{S}(AX-B)_{*i}\|_2 \\
&\geq (1-\varepsilon^2) \sum_{\text{good } i} \|(AX-B)_{*i}\|_2 + (1-2\varepsilon) \sum_{\text{large } i} \|(AX-B)_{*i}\|_2 \\
&\geq (1-\varepsilon^2) \sum_{\text{good } i} \|(AX-B)_{*i}\|_2 + (1-2\varepsilon) \sum_{\text{bad } i} \|(AX-B)_{*i}\|_2 \\
&\quad - (1-2\varepsilon) \sum_{\text{small } i} \|(AX-B)_{*i}\|_2 \\
&\geq (1-2\varepsilon)\|AX-B\|_h - (1-2\varepsilon)4\varepsilon\Delta^* \\
&\geq (1-6\varepsilon)\|AX-B\|_h.
\end{aligned}$$

Thus, by a union bound, with failure probability $\leq \delta$, $\boldsymbol{S}$ is an affine 6ε-contraction for (A, B) with respect to $\|\cdot\|_h$. □

Lemma A.1.3 (Gaussian Matrices are Lopsided Embeddings). *Given arbitrary matrices A of rank k and B of any rank, a Gaussian matrix $\boldsymbol{S}$ with $\widetilde{O}(k/\varepsilon^4 + 1/\varepsilon^4\delta^2)$ rows is an ε-lopsided embedding for (A, B) with probability $\geq 1-\delta$.*

Proof. We now show that a Gaussian matrix, with small dimension equal to $\widetilde{O}(k/\varepsilon^4 + 1/\varepsilon^4\delta^2)$, satisfies all the sufficient conditions of Lemma A.1.2. Clearly, a Gaussian matrix with $O((k+\log(1/\delta))/\varepsilon^2)$ rows satisfies condition 1 and a Gaussian matrix with $O((k+\log(1/\delta\varepsilon))/\varepsilon^4)$ rows satisfies condition 2 [Woo14].

We now show that a Gaussian matrix with at least $O(1/\varepsilon^4)$ rows satisfies

$$\mathbf{E}[(\|\boldsymbol{S}y\|_2^2 - 1)^2] \leq \varepsilon^4$$

for any given unit vector y. If $\boldsymbol{S}$ is a Gaussian matrix of t rows with each entry drawn i.i.d. from $N(0, 1/t)$, then the entries of $\boldsymbol{S}y$ are each drawn i.i.d. from $N(0, \|y\|_2^2/t) = N(0, 1/t)$. Therefore,

$\|Sy\|_2^2 = Y_1^2 + \ldots + Y_t^2$, where $Y_i \sim N(0, 1/t)$, which gives

$$\begin{aligned}\mathbf{E}[(\|Sy\|_2^2 - 1)^2] &= \mathbf{E}[(Y_1^2 + \ldots + Y_t^2 - 1)^2] \\ &= t\,\mathbf{E}[Y_1^4] + 1 + 2\binom{t}{2}\mathbf{E}[Y_1^2 Y_2^2] - 2t\,\mathbf{E}[Y_1^2] = t\frac{3}{t^2} + 1 + 2\binom{t}{2}\frac{1}{t^2} - 2t\frac{1}{t} \\ &= 2/t.\end{aligned}$$

Thus, with $t \geq 1/\varepsilon^4$, we have that $\mathbf{E}[(\|Sy\|_2^2 - 1)^2] \leq \varepsilon^4$. By Lemma 28 of [CW15], we obtain that $\mathbf{E}[\max(\|Sy\|_2^4, 1)] \leq (1+\varepsilon^2)^2 \leq 1 + 3\varepsilon^2$. Now, by Holder's inequality,

$$\mathbf{E}[\max(\|Sy\|_2, 1)] \leq \mathbf{E}[\max(\|Sy\|_2, 1)^4]^{1/4} \leq (1 + 3\varepsilon^2)^{1/4} \leq 1 + (3/4)\varepsilon^2.$$

As $(\|Sy\|_2 - 1)_+ = \max(\|Sy\|_2, 1) - 1$, we obtain that $\mathbf{E}[(\|Sy\|_2 - 1)_+] \leq (3/4)\varepsilon^2$, which implies by scaling that for an arbitrary vector y,

$$\mathbf{E}[(\|Sy\|_2 - \|y\|_2)_+] \leq (3/4)\varepsilon^2\|y\|_2$$

which gives

$$\mathbf{E}[(\|SB^*\|_h - \|B^*\|_h)_+] \leq (3/4)\varepsilon^2\|B^*\|_h.$$

By Markov's inequality, with probability $\geq 1 - \delta/3$, $(\|SB^*\|_h - \|B^*\|_h)_+ \leq (9/4)(\varepsilon^2/\delta)\|B^*\|_h$ and hence, with probability $\geq 1 - \delta/3$, $\|SB^*\|_h \leq (1 + (9/4)(\varepsilon^2/\delta))\|B^*\|_h$. Thus, a Gaussian matrix with $m = O(1/\varepsilon^4\delta^2)$ rows satisfies that with probability $\geq 1 - \delta/3$ that

$$\|SB^*\|_h \leq (1 + \varepsilon^2)\|B^*\|_h. \qquad \square$$

A.2 Utilizing Sampling-based ℓ_1 Embeddings

Let A be a matrix that has r columns. Suppose $\boldsymbol{L}$ is a random matrix such that with probability $\geq 9/10$, simultaneously for all vectors y,

$$\alpha\|Ay\|_1 \leq \|\boldsymbol{L}Ay\|_1 \leq \beta\|Ay\|_1.$$

Assume the above event holds. Let X be an arbitrary matrix with t columns. We have that for a suitably scaled Gaussian matrix $\boldsymbol{G}$ with $\widetilde{O}(t/\varepsilon^2)$ columns, with probability $\geq 9/10$, simultaneously for all vectors $x \in \mathbb{R}^t$, $\|x^{\mathrm{T}}\boldsymbol{G}\|_1 = (1 \pm \varepsilon)\|x\|_2$ [Mat13]. Thus, there exists a matrix M with $\widetilde{O}(t/\varepsilon^2)$ columns such that for all vectors $x \in \mathbb{R}^t$,

$$\|x^{\mathrm{T}}M\|_1 = (1 \pm \varepsilon)\|x\|_2.$$

Therefore,

$$\frac{1}{1+\varepsilon}\|AXM\|_{1,1} \leq \|AX\|_{1,2} = \frac{1}{1-\varepsilon}\|AXM\|_{1,1}$$

and

$$\frac{1}{1+\varepsilon}\|\boldsymbol{L}AXM\|_{1,1} \leq \|\boldsymbol{L}AX\|_{1,2} \leq \frac{1}{1-\varepsilon}\|\boldsymbol{L}AXM\|_{1,1}$$

Now, we upper bound $\|\boldsymbol{L}AX\|_{1,2}$.

$$\begin{aligned}\|\boldsymbol{L}AX\|_{1,2} &\leq \frac{1}{1-\varepsilon}\|\boldsymbol{L}AXM\|_{1,1} \leq \frac{1}{1-\varepsilon}\sum_j \|\boldsymbol{L}A(XM)_{*j}\|_1 \\ &\leq \frac{\beta}{1-\varepsilon}\sum_j \|A(XM)_{*j}\|_1 = \frac{\beta}{1-\varepsilon}\|AXM\|_{1,1} \leq \beta\frac{1+\varepsilon}{1-\varepsilon}\|AX\|_{1,2}.\end{aligned}$$

We now lower bound $\|\boldsymbol{L}AX\|_{1,2}$ similarly.

$$\begin{aligned}\|\boldsymbol{L}AX\|_{1,2} &\geq \frac{1}{1+\varepsilon}\|\boldsymbol{L}AXM\|_{1,1} = \frac{1}{1+\varepsilon}\sum_j \|\boldsymbol{L}A(XM)_{*j}\|_1 \\ &\geq \frac{\alpha}{1+\varepsilon}\sum_j \|A(XM)_{*j}\|_1 = \frac{\alpha}{1+\varepsilon}\|AXM\|_{1,1} \geq \alpha\frac{1-\varepsilon}{1+\varepsilon}\|AX\|_{1,2}.\end{aligned}$$

By picking appropriate ε, we conclude that for any matrix X,

$$\frac{\alpha}{2}\|AX\|_{1,2} \leq \|\boldsymbol{L}AX\|_{1,2} \leq 2\beta\|AX\|_{1,2}. \tag{A.2}$$

Lemma A.2.1. *If $\boldsymbol{S}^{\mathrm{T}}$ is a random Gaussian matrix with $O(k)$ columns such that with probability $\geq 9/10$,*

$$\min_{\text{rank-}k\ X} \|A\boldsymbol{S}^{\mathrm{T}}X - A\|_{1,2} \leq (3/2) \min_{\text{rank-}k\ X} \|AX - A\|_{1,2},$$

and if $\boldsymbol{L}$ is a random matrix drawn from a distribution such that with probability $\geq 9/10$ over the draw of matrix $\boldsymbol{L}$,

$$\alpha\|A\boldsymbol{S}^{\mathrm{T}}y\|_1 \leq \|\boldsymbol{L}A\boldsymbol{S}^{\mathrm{T}}y\|_1 \leq \beta\|A\boldsymbol{S}^{\mathrm{T}}y\|_1$$

for all vectors y and

$$\mathbf{E}_{\boldsymbol{L}}[\|\boldsymbol{L}M\|_{1,2}] = \|M\|_{1,2}$$

for any matrix M, then with probability $\geq 3/5$, all matrices X such that $\|\boldsymbol{L}A\boldsymbol{S}^{\mathrm{T}}X - \boldsymbol{L}A\|_{1,2} \leq 10 \cdot \mathrm{SubApx}_{k,1}(A)$ satisfy

$$\|A\boldsymbol{S}^{\mathrm{T}}X - A\|_{1,2} \leq (2 + 40/\alpha)\,\mathrm{SubApx}_{k,1}(A).$$

Proof. Let $X_1 = \arg\min_{\text{rank-}k\ X} \|A\boldsymbol{S}^{\mathrm{T}}X - A\|_{1,2}$. With probability $\geq 9/10$, we have that $\|A\boldsymbol{S}^{\mathrm{T}}X_1 -$

$A\|_{1,2} \le (3/2)\mathrm{SubApx}_{k,1}(A)$. By a Markov bound, we obtain that with probability $\ge 4/5$, $\|\boldsymbol{LAS}^{\mathrm{T}}X_1 - \boldsymbol{L}A\|_{1,2} \le 10 \cdot \mathrm{SubApx}_{k,1}(A)$. Assume this event holds. For any matrix X,

$$\|\boldsymbol{LAS}^{\mathrm{T}}X - \boldsymbol{L}A\|_{1,2} \ge \|\boldsymbol{LAS}^{\mathrm{T}}X - \boldsymbol{LAS}^{\mathrm{T}}X_1\|_{1,2} - \|\boldsymbol{LAS}^{\mathrm{T}}X_1 - \boldsymbol{L}A\|_{1,2}.$$

We have

$$\|\boldsymbol{LAS}^{\mathrm{T}}X - \boldsymbol{L}A\|_{1,2} \ge \|\boldsymbol{LAS}^{\mathrm{T}}X - \boldsymbol{LAS}^{\mathrm{T}}X_1\|_{1,2} - 10 \cdot \mathrm{SubApx}_{k,1}(A).$$

From (A.2), we have

$$\begin{aligned}
\|\boldsymbol{LAS}^{\mathrm{T}}X - \boldsymbol{L}A\|_{1,2} &\ge \frac{\alpha}{2}\|A\boldsymbol{S}^{\mathrm{T}}X - A\boldsymbol{S}^{\mathrm{T}}X_1\|_{1,2} - 10 \cdot \mathrm{SubApx}_{k,1}(A) \\
&\ge \frac{\alpha}{2}\|A\boldsymbol{S}^{\mathrm{T}}X - A\|_{1,2} - \frac{\alpha}{2}\|A\boldsymbol{S}^{\mathrm{T}}X_1 - A\|_{1,2} - 10 \cdot \mathrm{SubApx}_{k,1}(A) \\
&\ge \frac{\alpha}{2}\|A\boldsymbol{S}^{\mathrm{T}}X - A\|_{1,2} - (3\alpha/4 + 10) \cdot \mathrm{SubApx}_{k,1}(A).
\end{aligned}$$

Thus, for any matrix X of rank r, if $\|A\boldsymbol{S}^{\mathrm{T}}X - A\|_{1,2} > (2/\alpha)(20 + 3\alpha/4) \cdot \mathrm{SubApx}_{k,1}(A)$, then $\|\boldsymbol{LAS}^{\mathrm{T}}X - \boldsymbol{L}A\|_{1,2} > 10 \cdot \mathrm{SubApx}_{k,1}(A)$. □

A.3 Finding Best Solution Among Candidate Solutions

Algorithm 4.1 finds candidate solutions $\hat{X}^{(1)}, \ldots, \hat{X}^{(t)}$ for $t = O(\log(1/\delta))$ and returns the best candidate solution $\hat{X}$ among $\hat{X}^{(1)}, \ldots, \hat{X}^{(t)}$ that minimizes the cost

$$\|A(I - BB^{\mathrm{T}})(I - \hat{X}\hat{X}^{\mathrm{T}})\|_{1,2}. \tag{A.3}$$

The proof of Theorem 4.4.3 shows that, for all $i = 1, \ldots, t$, with probability $\ge 3/5$, $\|A(I - BB^{\mathrm{T}})(I - \hat{X}^{(i)}(\hat{X}^{(i)})^{\mathrm{T}})\|_{1,2} \le O(1) \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}}))$. Therefore, with probability $\ge 1 - \delta/2$

$$\min_i \|A(I - BB^{\mathrm{T}})(I - \hat{X}(\hat{X})^{\mathrm{T}})\|_{1,2} \le O(1) \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})) \tag{A.4}$$

i.e., with probability $\ge 1 - \delta$, there is a solution $\hat{X}^{(i)}$ among the t potential solutions that has a cost at most $O(1) \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}}))$. We first compute

$$\mathrm{apx}_i = \|A(I - BB^{\mathrm{T}})(I - \hat{X}^{(i)}(\hat{X}^{(i)})^{\mathrm{T}})\boldsymbol{G}\|_{1,2}$$

where $\boldsymbol{G}$ is a scaled Gaussian matrix with $O(\log(n/\delta))$ columns. Values of apx_j for all $j \in [t]$ can be computed in time $\widetilde{O}((\mathrm{nnz}(A) + (n + d)\,\mathrm{poly}(k/\varepsilon)) \cdot \log(1/\delta))$. We have using the union bound that, with probability $\geq 1 - \delta/2$, for all $j \in [n]$ and $i \in [t]$ that

$$\|A_{j*}(I - BB^{\mathrm{T}})(I - \hat{X}^{(i)}(\hat{X}^{(i)})^{\mathrm{T}})\boldsymbol{G}\|_2 = (1/2, 3/2)\|A_{j*}(I - BB^{\mathrm{T}})(I - \hat{X}^{(i)}(\hat{X}^{(i)})^{\mathrm{T}})\|_2. \tag{A.5}$$

Therefore, with probability $\geq 1 - \delta/2$, for all $i \in [t]$,

$$\mathrm{apx}_i \in (1/2, 3/2)\|A(I - BB^{\mathrm{T}})(I - \hat{X}^{(i)}(\hat{X}^{(i)})^{\mathrm{T}})\|_{1,2}. \tag{A.6}$$

Let $\widetilde{i} = \arg\min_{i \in [t]} \mathrm{apx}_i$ and $i^* = \arg\min_{i \in [t]} \|A(I - BB^{\mathrm{T}})(I - \hat{X}^{(i)}(\hat{X}^{(i)})^{\mathrm{T}})\|_{1,2}$. By a union bound, with probability $\geq 1 - \delta$

$$\begin{aligned}
\|A(I - BB^{\mathrm{T}})(I - \hat{X}^{(\widetilde{i})}(\hat{X}^{(\widetilde{i})})^{\mathrm{T}})\|_{1,2} &\leq 2\mathrm{apx}_{\widetilde{i}} \\
&\leq 2\mathrm{apx}_{i^*} \\
&\leq 4\|A(I - BB^{\mathrm{T}})(I - \hat{X}^{(i^*)}(\hat{X}^{(i^*)})^{\mathrm{T}})\|_{1,2} \\
&\leq O(1) \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}})).
\end{aligned}$$

Thus, Algorithm 4.1, with probability $\geq 1 - \delta$, returns a subspace that has cost at most $O(\sqrt{k}) \cdot \mathrm{SubApx}_{k,1}(A(I - BB^{\mathrm{T}}))$ and has a running time of $\widetilde{O}((\mathrm{nnz}(A) + (n + d)\,\mathrm{poly}(k/\varepsilon)) \cdot \log(1/\delta))$.

Appendix B

Deferred Proofs from Chapter 6

B.1 Omitted Proofs from Section 6.3

B.1.1 Proof of Theorem 6.3.1

Proof. Without loss of generality, we prove the theorem assuming A has orthonormal columns. Thus, $U = A$. Let X be an arbitrary matrix such that $\|UX - B\|_2 < 1$. We will give a series of statements equivalent to $\|UX - B\|_2 < 1$ that prove the theorem. Using the fact that for any matrix A, $\|A\|_2 < 1$ if and only if $A^{\mathrm{T}}A \prec I$, we obtain the equivalent statement

$$(UX - B)^{\mathrm{T}}(UX - B) \prec I.$$

Writing B as $UU^{\mathrm{T}}B + (I - UU^{\mathrm{T}})B$, we get another equivalent statement

$$(UX - UU^{\mathrm{T}}B)^{\mathrm{T}}(UX - UU^{\mathrm{T}}B) \prec I - B^{\mathrm{T}}(I - UU^{\mathrm{T}})B = I - \Delta.$$

As the LHS of the above relation is a positive semi-definite matrix, we obtain that $I - \Delta \succ 0$ and hence is invertible. Thus, the above condition can be equivalently written as

$$(I - \Delta)^{-1/2}(UX - UU^{\mathrm{T}}B)^{\mathrm{T}}(UX - UU^{\mathrm{T}}B)(I - \Delta)^{-1/2} \prec I.$$

Using the fact that Δ is symmetric and $U^{\mathrm{T}}U = I$, we get that the above condition is the same as

$$\|X(I - \Delta)^{-1/2} - U^{\mathrm{T}}B(I - \Delta)^{-1/2}\|_2 < 1.$$

Thus, we obtain that in the case that $\|(I - UU^{\mathrm{T}})B\|_2 < 1$, for an arbitrary matrix X, the condition that $\|UX - B\|_2 < 1$ is equivalent to $\|X(I-\Delta)^{-1/2} - U^{\mathrm{T}}B(I-\Delta)^{-1/2}\|_2 < 1$. Let $\hat{B} \coloneqq U^{\mathrm{T}}B(I-\Delta)^{-1/2}$. It is easy to see that $X = [\hat{B}]_{\mathsf{sve}(\hat{B})}(I - \Delta)^{1/2}$ satisfies $\|UX - B\|_2 < 1$ and that any matrix X that satisfies $\|UX - B\|_2 < 1$ must have rank at least $\mathrm{sve}(\hat{B})$. All that remains to show is that $\mathrm{sve}(\hat{B}) = \mathrm{sve}(B)$. We will show that $k^-(I - \hat{B}^\mathrm{T}\hat{B}) = k^-(I - B^\mathrm{T}B)$, which completes the proof:

$$\begin{aligned}
I - \hat{B}^\mathrm{T}\hat{B} &= I - (I-\Delta)^{-1/2}B^\mathrm{T}UU^\mathrm{T}B(I-\Delta)^{-1/2} \\
&= I - (I-\Delta)^{-1/2}(B^\mathrm{T}B - \Delta)(I-\Delta)^{-1/2} \\
&= I - (I-\Delta)^{-1/2}(B^\mathrm{T}B - I + I - \Delta)(I-\Delta)^{-1/2} \\
&= I - I + (I-\Delta)^{-1/2}(I - B^\mathrm{T}B)(I-\Delta)^{-1/2} \\
&= (I-\Delta)^{-1/2}(I - B^\mathrm{T}B)(I-\Delta)^{-1/2}.
\end{aligned}$$

Thus, $k^-(I - \hat{B}^\mathrm{T}\hat{B}) = k^-((I-\Delta)^{-1/2}(I - B^\mathrm{T}B)(I-\Delta)^{-1/2})$. By Sylvester's law of inertia [Car17, p313], $k^-((I-\Delta)^{-1/2}(I - B^\mathrm{T}B)(I-\Delta)^{-1/2}) = k^-(I - B^\mathrm{T}B)$. Therefore,

$$\mathrm{sve}(\hat{B}) = k^-(I - \hat{B}^\mathrm{T}\hat{B}) = k^-(I - B^\mathrm{T}B) = \mathrm{sve}(B).$$

Thus, $\mathrm{sve}(B)$ is the optimum value for (6.1) if it is feasible. □

B.2 Omitted Proofs from Section 6.4

B.2.1 Proof of Lemma 6.4.1

Proof. The proof of this lemma is very similar to the proof of Theorem 6.3.1. Suppose there exists a rank-k matrix X such that $\|UX - B\|_2 < \beta$. We already have $\beta > \|(I - UU^\mathrm{T})B\|_2$. The statement $\|UX - B\|_2 < \beta$ implies that

$$(UX - B)^\mathrm{T}(UX - B) \preceq \beta^2 I.$$

We can write $B = UU^\mathrm{T}B + (I - UU^\mathrm{T})B$ and obtain that for any matrix X, $(UX - B)^\mathrm{T}(UX - B) = (UX - UU^\mathrm{T}B)^\mathrm{T}(UX - UU^\mathrm{T}B) + \Delta$, which implies that

$$(UX - UU^\mathrm{T}B)^\mathrm{T}(UX - UU^\mathrm{T}B) \preceq \beta^2 I - \Delta.$$

As $\|\Delta\|_2 = \|(I - UU^\mathrm{T})B\|_2^2 < \beta^2$, $\beta^2 I - \Delta$ is invertible, which implies that

$$(\beta^2 I - \Delta)^{-1/2}(UX - UU^\mathrm{T}B)^\mathrm{T}(UX - UU^\mathrm{T}B)(\beta^2 I - \Delta)^{-1/2} \preceq I.$$

Thus, we have $\|(UX - UU^{\mathrm{T}}B)(\beta^2 I - \Delta)^{-1/2}\|_2 = \|X(\beta^2 I - \Delta)^{-1/2} - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2$ is less than or equal to 1. As X is a matrix of rank k, the matrix $X(\beta^2 I - \Delta)^{-1/2}$ also has rank k. Therefore,

$$\begin{aligned}\sigma_{k+1}(U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}) &= \|[U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}]_k - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2 \\ &\leq \|X(\beta^2 I - \Delta)^{-1/2} - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2 \\ &\leq 1.\end{aligned}$$

□

B.2.2 Proof of Lemma 6.4.2

Proof. Suppose Y is a rank k matrix such that $\|Y - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2 \leq 1 + \varepsilon$. Then we have $\|Y(\beta^2 I - \Delta)^{1/2}(\beta^2 I - \Delta)^{-1/2} - U^{\mathrm{T}}B(\beta^2 I - \Delta)^{-1/2}\|_2 \leq 1 + \varepsilon$ and therefore

$$(\beta^2 I - \Delta)^{-1/2}(Y(\beta^2 I - \Delta)^{1/2} - U^{\mathrm{T}}B)^{\mathrm{T}}(Y(\beta^2 I - \Delta)^{1/2} - U^{\mathrm{T}}B)(\beta^2 I - \Delta)^{-1/2} \preceq (1 + \varepsilon)^2 I.$$

Multiplying the above relation on both sides with $(\beta^2 I - \Delta)^{1/2}$ on the left and the right, we obtain

$$(Y(\beta^2 I - \Delta)^{1/2} - U^{\mathrm{T}}B)^{\mathrm{T}}(Y(\beta^2 I - \Delta)^{1/2} - U^{\mathrm{T}}B) \preceq (1 + \varepsilon)^2(\beta^2 I - \Delta).$$

Using $U^{\mathrm{T}}U = I$ and adding Δ to both sides, we conclude that

$$\|UY(\beta^2 I - \Delta)^{1/2} - B\|_2 \leq \sqrt{\|(1 + \varepsilon)^2\beta^2 I\|_2} \leq (1 + \varepsilon)\beta.$$

Now Y is a matrix that has rank at most k. We also have $\|UYZ - B\|_2 \geq \|UY(UY)^+B - B\|_2$ for any matrix Z. Therefore, $\|UY(UY)^+B - B\|_2 \leq \|UY(\beta^2 I - \Delta)^{1/2} - B\|_2 \leq (1 + \varepsilon)\beta$. □

B.3 Omitted Proofs from Section 6.5

B.3.1 Error in Computing Krylov Subspace

Given a matrix $M \in \mathbb{R}^{n \times d}$, an integer $k \leq d$ and an odd integer $q \geq 0$, the Krylov subspace is defined by

$$K = [(MM^{\mathrm{T}})^{(q-1)/2}MG,\ (MM^{\mathrm{T}})^{(q-3)/2}MG,\ \cdots,\ (MM^{\mathrm{T}})^1 MG,\ MG]$$

where G is a $d \times k$ matrix with i.i.d. normal entries. Using the algorithm to approximately multiply a vector with the matrices M and M^{T}, we compute an approximation to the matrix K defined above. For any vector v, define $(MM^{\mathrm{T}})^{\circ 0}v := v$ and for $i > 0$, define $(MM^{\mathrm{T}})^{\circ i}v := M \circ (M^{\mathrm{T}} \circ ((MM^{\mathrm{T}})^{\circ(i-1)}v))$ (recall $M \circ v$ is the approximation to Mv computed by the oracle). The notation is similarly extended to define $(MM^{\mathrm{T}})^{\circ i}G$ for a matrix G. Now we define the matrix

$$K' = [(MM^{\mathrm{T}})^{\circ(q-1)/2}M \circ G,\ (MM^{\mathrm{T}})^{\circ(q-3)/2}M \circ G,\ \cdots,\ (MM^{\mathrm{T}})^{\circ 1}M \circ G,\ M \circ G].$$

We now bound $\|K - K'\|_{\mathrm{F}}$ and the time required to compute K' using the following lemma.

Lemma B.3.1. *For any matrix $M \in \mathbb{R}^{n\times d}$, matrix $G \in \mathbb{R}^{d\times k}$ and an odd integer q, let $\Delta_{i,G} \coloneqq (MM^{\mathrm{T}})^{(i-1)/2}MG - (MM^{\mathrm{T}})^{\circ(i-1)/2}M \circ G$ and matrices $K, K' \in \mathbb{R}^{n\times qk}$ be as defined above. Then*

$$E_{i,G} \coloneqq \|\Delta_{i,G}\|_{\mathrm{F}} \le 8\varepsilon_\circ(2^{i/2}\|M\|_2^i\|G\|_{\mathrm{F}})$$

for $i = 1, 3, 5, \ldots, q$ and $\|K - K'\|_{\mathrm{F}} \le O(\varepsilon_\circ\|G\|_{\mathrm{F}}\|M\|_2^{q+1}2^{(q+1)/2})$. The matrix K' can be computed in $O(T(\varepsilon_\circ)qk)$ time.

Proof. For an arbitrary vector v and i odd, let $\Delta_i \coloneqq (MM^{\mathrm{T}})^{(i-1)/2}Mv - (MM^{\mathrm{T}})^{\circ(i-1)/2}M \circ v$. Let $E_i = \|\Delta_i\|_2$. We have $E_1 = \|\Delta_1\|_2 = \|Mv - M \circ v\|_2 \le \|M\|_2\|v\|_2$. We now define a recurrence relation between E_i and E_{i-2} and then bound E_i using this recurrence. We have

$$\begin{aligned}
\Delta_i &= (MM^{\mathrm{T}})^{(i-1)/2}Mv - (MM^{\mathrm{T}})^{\circ(i-1)/2}M \circ v\\
&= (MM^{\mathrm{T}})(MM^{\mathrm{T}})^{(i-3)/2}Mv - (MM^{\mathrm{T}})^{\circ 1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v\\
&= (MM^{\mathrm{T}})[(MM^{\mathrm{T}})^{(i-3)/2}Mv - (MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v]\\
&\quad + [(MM^{\mathrm{T}})^{1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v - (MM^{\mathrm{T}})^{\circ 1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v]\\
&= (MM^{\mathrm{T}})\Delta_{i-2} + [(MM^{\mathrm{T}})^{1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v - (MM^{\mathrm{T}})^{\circ 1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v].
\end{aligned}$$

Therefore, by the triangle inequality of $\|\cdot\|_2$,

$$\begin{aligned}
E_i &\le \|MM^{\mathrm{T}}\Delta_{i-2}\|_2 + \|(MM^{\mathrm{T}})^{1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v - (MM^{\mathrm{T}})^{\circ 1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v\|_2\\
&\le \|M\|_2^2E_{i-2} + \|(MM^{\mathrm{T}})^{1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v - (MM^{\mathrm{T}})^{\circ 1}(MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v\|_2.
\end{aligned}$$

Let $v' \coloneqq (MM^{\mathrm{T}})^{\circ(i-3)/2}M \circ v$. We now bound $\|MM^{\mathrm{T}}v' - (MM^{\mathrm{T}})^{\circ 1}v'\|_2$:

$$\begin{aligned}
\|MM^{\mathrm{T}}v' - (MM^{\mathrm{T}})^{\circ 1}v'\|_2 &= \|MM^{\mathrm{T}}v' - M \circ (M \circ v')\|_2\\
&\le \|MM^{\mathrm{T}}v' - M(M^{\mathrm{T}} \circ v')\|_2 + \|M(M^{\mathrm{T}} \circ v') - M \circ (M^{\mathrm{T}} \circ v')\|_2\\
&\le \|M\|_2\|M^{\mathrm{T}}v' - M^{\mathrm{T}} \circ v'\|_2 + \varepsilon_\circ\|M\|_2\|M^{\mathrm{T}} \circ v'\|_2\\
&\le \varepsilon_\circ\|M\|_2^2\|v'\|_2 + \varepsilon_\circ\|M\|_2(\varepsilon_\circ\|M\|_2\|v'\|_2 + \|M^{\mathrm{T}}v'\|_2)\\
&\le 3\varepsilon_\circ\|M\|_2^2\|v'\|_2.
\end{aligned}$$

As $v' = (MM^{\mathrm{T}})^{(i-3)/2}Mv - \Delta_{i-2}$, we get $\|v'\|_2 \le \|(MM^{\mathrm{T}})^{(i-3)/2}Mv\|_2 + \|\Delta_{i-2}\|_2 \le \|M\|_2^{i-2}\|v\|_2 + E_{i-2}$. Therefore, we finally obtain that

$$\begin{aligned}
E_i \le \|M\|_2^2E_{i-2} + 3\varepsilon_\circ\|M\|_2^2\|v'\|_2 &\le \|M\|_2^2E_{i-2} + 3\varepsilon_\circ\|M\|_2^2(\|M\|_2^{i-2}\|v\|_2 + E_{i-2})\\
&\le (1 + 3\varepsilon_\circ)\|M\|_2^2E_{i-2} + 3\varepsilon_\circ\|M\|_2^i\|v\|_2.
\end{aligned}$$

Solving this recurrence relation we obtain that

$$\begin{aligned}E_i &\le (1+3\varepsilon_\circ)^{(i-1)/2}\|M\|_2^{i-1}E_1 + (1+(1+3\varepsilon_\circ)+\cdots+(1+3\varepsilon_\circ)^{(i-3)/2})(3\varepsilon_\circ\|M\|_2^i\|v\|_2)\\ &\le \varepsilon_\circ(1+2^{(i-1)/2}(3\varepsilon_\circ))\|M\|_2^i\|v\|_2 + 2^{(i-1)/2}(3\varepsilon_\circ)\|M\|_2^i\|v\|_2\\ &\le 8(\varepsilon_\circ 2^{i/2}\|M\|_2^i\|v\|_2).\end{aligned}$$

In the above inequalities, we used the standard inequality $(1+x)^n \le 1+2^n x$ if $0 \le x \le 1$. Thus, for any arbitrary vector v, $\|(MM^{\mathrm{T}})^{\circ(i-1)/2}M \circ v - (MM^{\mathrm{T}})^{(i-1)/2}Mv\|_2 \le 8\varepsilon_\circ 2^{i/2}\|M\|_2^i\|v\|_2$ and therefore for the Gaussian matrix G,

$$E_{i,G} = \|(MM^{\mathrm{T}})^{\circ(i-1)/2}M \circ G - (MM^{\mathrm{T}})^{(i-1)/2}MG\|_{\mathrm{F}} \le 8\varepsilon_\circ 2^{i/2}\|M\|_2^i\|G\|_{\mathrm{F}}.$$

We then have that $\|K-K'\|_{\mathrm{F}} \le O(\varepsilon_\circ\|G\|_{\mathrm{F}}\|M\|_2^{q+1}2^{(q+1)/2})$. In computing the matrix K' we make $O(qk)$ calls to each of the oracles and therefore take $O(T(\varepsilon_\circ)qk)$ time. □

Musco and Musco [MM15] consider a polynomial $p(x)$ such that the column space of the matrix $p(M)G$ is spanned by K. They then argue that the column span of $p(M)G$ is a "good" k-dimensional subspace to project M onto and then conclude that the best rank k approximation of M inside the span of K satisfies (6.2). Although we have an upper bound on $\|K-K'\|_{\mathrm{F}}$ from the above lemma, we cannot directly argue that the best rank k approximation of M inside K' satisfies the guarantee of (6.2), as the matrix K might be very poorly conditioned.

To overcome this issue, we first show that the matrix $p(M)G$ has a bounded condition number with $O(1)$ probability and that K' spans a matrix Apx that is close to $p(M)G$. We then show that the span of the matrix Apx is a good subspace to project the matrix M onto and then conclude that the best rank k approximation of M inside the span of K' satisfies (6.2).

B.3.2 Condition Number of the matrix $p(M)G$ and existence of good rank k subspace inside an approximate Krylov Subspace

Throughout this section let $\alpha = \sigma_{k+1}(M)$ and $\gamma = \varepsilon/2$. Let q be an odd integer and $T(x)$ be the degree q Chebyshev polynomial. Define

$$p(x) \coloneqq (1+\gamma)\alpha\frac{T(x/\alpha)}{T(1+\gamma)}. \tag{B.1}$$

The following lemma bounds $\sigma_1(p(M))/\sigma_{k+1}(p(M))$ which lets us bound $\kappa(p(M)G)$.

Lemma B.3.2. *If $M \in \mathbb{R}^{n\times d}$ is a matrix such that $\sigma_1(M)/\sigma_{k+1}(M) = \kappa$, then*

$$\sigma_1(p(M))/\sigma_{k+1}(p(M)) \le (3\kappa)^q.$$

First, we have the following lemma that shows that $T(x) \geq 1$ for all $x \geq 1$ for the Chebyshev Polynomial T of any degree d.

Lemma B.3.3. *If $T_d(x)$ is the degree d Chebyshev Polynomial, then for all $d \geq 0$ and for all $x \geq 1$, $T_{d+1}(x) \geq T_d(x) \geq 1$.*

Proof. We prove the theorem using induction on the degree d. We have $T_0(x) = 1$ and $T_1(x) = x$. Thus, $T_1(x) \geq T_0(x) \geq 1$ for $x \geq 1$. Assume that for all $d < n$ and $x \geq 1$, $T_{d+1}(x) \geq T_d(x) \geq 1$. If we now prove that $T_{n+1}(x) \geq T_n(x) \geq 1$, we are done by induction.

We have $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) = T_n(x) + [T_n(x) - T_{n-1}(x)] + (2x-2)T_n(x)$. As $x \geq 1$ and by the induction hypothesis $T_n(x) \geq T_{n-1}(x) \geq 1$, we obtain that $T_{n+1}(x) \geq T_n(x) \geq 1$. Thus, for all $d \geq 0$ and $x \geq 1$, $T_{d+1}(x) \geq T_d(x) \geq 1$. □

Recall $p(x) = (1+\gamma)\alpha\frac{T(x/\alpha)}{T(1+\gamma)} = (1+\varepsilon/2)\sigma_{k+1}\frac{T(x/\alpha)}{T(1+\gamma)}$.

Lemma B.3.4. *If $x \geq \alpha > 0$, then $p(x) \leq (1+\gamma)\alpha\frac{3^q(x/\alpha)^q}{T(1+\gamma)}$.*

Proof. By a standard property, the sum of absolute values of coefficients of the degree-q Chebyshev polynomial is bounded above by 3^q. Thus, $T(x/\alpha) = \sum_{i=1}^q T_i(x/\alpha)^i \leq \sum_{i=1}^q |T_i|(x/\alpha)^i \leq (x/\alpha)^q \sum_{i=1}^q |T_i| \leq 3^q(x/\alpha)^q$, where we use the fact that $(x/\alpha) \geq 1$. Therefore, $p(x) = (1+\gamma)\alpha T(x/\alpha)/T(1+\gamma) \leq (1+\gamma)\alpha 3^q(x/\alpha)^q/T(1+\gamma)$. □

Proof of Lemma B.3.2. We bound $\sigma_1(p(M))$ and $\sigma_{k+1}(p(M))$, and then infer an upper bound on $\frac{\sigma_1(p(M))}{\sigma_{k+1}(p(M))}$. Let $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d \geq 0$ be the singular values of the matrix M. Then $|p(\sigma_1)|, |p(\sigma_2)|, \ldots, |p(\sigma_d)|$ are the singular values of the matrix $p(M)$. Consider any $i \leq k+1$. We have $\sigma_i \geq \sigma_{k+1} = \alpha$. Therefore,

$$p(\sigma_i) = (1+\gamma)\sigma_{k+1}\frac{T(\sigma_i/\sigma_{k+1})}{T(1+\gamma)} \geq \frac{(1+\gamma)\sigma_{k+1}}{T(1+\gamma)}.$$

Here we use Lemma B.3.3 to lower bound the value of $T(\sigma_i/\sigma_{k+1})$ by 1. Therefore, at least $k+1$ singular values of $p(M)$ are at least $\frac{(1+\gamma)\sigma_{k+1}}{T(1+\gamma)}$, which implies $\sigma_{k+1}(p(M)) \geq \frac{(1+\gamma)\sigma_{k+1}}{T(1+\gamma)}$.

Now for any $i \leq k+1$, $p(\sigma_i) \leq (1+\gamma)\sigma_{k+1}(3^q\kappa^q)/T(1+\gamma)$ by Lemma B.3.4. For any $i \geq k+1$, we have that $\sigma_i \leq \sigma_{k+1}$ and $|p(\sigma_i)| = (1+\gamma)\sigma_{k+1}|T(\sigma_i/\sigma_{k+1})|/T(1+\gamma) \leq (1+\gamma)\sigma_{k+1}/T(1+\gamma)$ by a well known property of Chebyshev polynomials that $|T(x)| \leq 1$ for all $x \in [-1,1]$. Therefore,

$$\|p(M)\|_2 = \sigma_1(p(M)) = \max_i |p(\sigma_i(M))| \leq (1+\gamma)\sigma_{k+1}\frac{3^q\kappa^q}{T(1+\gamma)}. \tag{B.2}$$

Thus, $\sigma_1(p(M))/\sigma_{k+1}(p(M)) \leq 3^q\kappa^q$. □

We now bound the condition number of the matrix $p(M)G$ where G is a Gaussian matrix with k columns. We use results from [RV10] to bound the maximum and minimum singular values of G with $O(1)$ probability and then use the above lemma to obtain bounds on the extreme singular values of $p(M)G$.

Lemma B.3.5. *If $G \in \mathbb{R}^{d\times k}$ is a matrix of i.i.d. normal entries and $M \in \mathbb{R}^{n\times d}$ is a matrix such that $\sigma_1(M)/\sigma_{k+1}(M) = \kappa$, then with probability $\geq 4/5$,*

$$\kappa(p(M)G) = \sigma_{\max}(p(M)G)/\sigma_{\min}(p(M)G) \leq C\sqrt{dk}3^q\kappa^q,$$

for an absolute constant $C > 0$.

Lemma B.3.6. *If $A \in \mathbb{R}^{n\times d}$ is a matrix with $\sigma_1(A)/\sigma_k(A) \leq \kappa_1$ and $G \in \mathbb{R}^{d\times k}$ is a matrix with i.i.d. normal entries, then for d greater than a constant, with probability $\geq 4/5$, the matrix AG has full rank and has $\sigma_1(AG)/\sigma_k(AG) \leq C\sqrt{dk}(\sigma_1(A)/\sigma_k(A))$, where $C > 0$ is an absolute constant.*

Proof. Let $A = U\Sigma V^{\mathrm{T}}$ be the singular value decomposition of A with $U \in \mathbb{R}^{n\times n}, \Sigma \in \mathbb{R}^{n\times d}$ and $V^{\mathrm{T}} \in \mathbb{R}^{d\times d}$. Let $G' = V^{\mathrm{T}}G$. As rows of V^{T} are orthonormal and entries of G are i.i.d. normal random variables, we obtain that G' is also a matrix of i.i.d. normal random variables of size $d\times k$. Let $\Sigma_k \in \mathbb{R}^{k\times k}$ be the diagonal matrix formed by the first k singular values. We first bound $\sigma_{\min}(AG) = \sigma_{\min}(U\Sigma V^{\mathrm{T}}G) = \sigma_{\min}(\Sigma\cdot G')$. Assuming that $\sigma_k \geq 0$, we note that $\mathrm{rank}(\Sigma_k\cdot G'_k) = \mathrm{rank}(\Sigma\cdot G') = k$ where G'_k is the $k\times d$ matrix formed by the first k rows of the matrix G'. Hence,

$$\sigma_{\min}(\Sigma\cdot G') \geq \sigma_{\min}(\Sigma_k\cdot G'_k) \geq \sigma_k(A)\cdot\sigma_{\min}(G'_k).$$

We have that

$$\mathbf{Pr}[\sigma_{\min}(G') \leq \frac{1}{40C\cdot\sqrt{k}}] \leq \left(\frac{1}{20}\right) + e^{-ck}$$

for some absolute constants c and C by Theorem 1.1 of [RV09]. Thus, for large enough d, with probability $\geq 9/10$, we have

$$\sigma_{\min}(G') \geq \frac{1}{40C\cdot\sqrt{k}}.$$

Thus, $\sigma_{\min}(A\cdot G) \geq \Omega(1/\sqrt{k})$ with probability $\geq 9/10$. Similarly, for large enough d, we have with probability $\geq 9/10$ that $\sigma_{\max}(G') \leq D(\sqrt{d}+\sqrt{k})$ for an absolute constant D by Proposition 2.4 of [RV10] and therefore $\max_{x:\|x\|_2=1}\|AGx\|_2 = \max_{x:\|x\|_2=1}\|\Sigma G'x\|_2 \leq D\sigma_1(A)(\sqrt{d}+\sqrt{k})$. Therefore, with probability $\geq 4/5$,

$$\kappa(AG) = \frac{\sigma_{\max}(AG)}{\sigma_{\min}(AG)} \leq 40CD\frac{\sigma_1(A)}{\sigma_k(A)}\sqrt{dk}.$$ □

Proof of Lemma B.3.5. Using the above lemma, we have that with probability $\geq 4/5$,

$$\kappa(p(M)G) = \frac{\sigma_{\max}(p(M)G)}{\sigma_{\min}(p(M)G)} \leq C\sqrt{dk}\frac{\sigma_1(p(M))}{\sigma_k(p(M))} \leq C\sqrt{dk}\frac{\sigma_1(p(M))}{\sigma_{k+1}(p(M))}$$

for an absolute constant C. The last inequality follows from $\sigma_k(p(M)) \geq \sigma_{k+1}(p(M))$. From Lemma B.3.2, we have $\frac{\sigma_1(p(M))}{\sigma_{k+1}(p(M))} \leq 3^q\kappa^q$. Therefore, $\kappa(p(M)G) \leq Ck3^q\kappa^q$ with probability $\geq 4/5$. □

The bound on the condition number of $p(M)G$ enables us to conclude that if the Frobenius norm error between $p(M)G$ and a matrix Apx is small, then the projection matrices onto the column spaces of the matrices $p(M)G$ and Apx are close. Specifically, we use the following lemma.

Lemma B.3.7. *Let A and B be full column rank matrices such that $\|A-B\|_2 \le \delta\|A\|_2$. Let $\kappa(A)$ denote the condition number of the matrix A i.e., $\kappa(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$. Let U and V denote an orthonormal basis for matrices A and B, respectively. If $\delta \le 1/(2\kappa(A)) \le 1$, then $\|AA^+ - BB^+\|_2 = \|UU^{\mathrm{T}} - VV^{\mathrm{T}}\|_2 \le 20\delta\kappa(A)^4$.*

Proof. As A and B are full rank matrices, we have $A^+ = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$ and $B^+ = (B^{\mathrm{T}}B)^{-1}B^{\mathrm{T}}$. Let $A - B = \Delta$. We have $\|\Delta\|_2 \le \delta\|A\|_2$. We first have

$$\begin{aligned}\|AA^+ - BB^+\|_2 &= \|AA^+ - (A-\Delta)B^+\|_2\\ &\le \|A\|_2\|A^+ - B^+\|_2 + \|\Delta\|_2\|B^+\|_2\\ &\le \|A\|_2\|A^+ - B^+\|_2 + \frac{\|\Delta\|_2}{\sigma_{\min}(B)}.\end{aligned}$$

Note that $A^{\mathrm{T}}A = (B+\Delta)^{\mathrm{T}}(B+\Delta) = B^{\mathrm{T}}B + \Delta^{\mathrm{T}}B + B^{\mathrm{T}}\Delta + \Delta^{\mathrm{T}}\Delta$. Now,

$$\begin{aligned}\|A^+ - B^+\|_2 &= \|(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} - (B^{\mathrm{T}}B)^{-1}B^{\mathrm{T}}\|_2\\ &= \|(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} - (B^{\mathrm{T}}B)^{-1}(A^{\mathrm{T}} - \Delta^{\mathrm{T}})\|_2\\ &\le \|(A^{\mathrm{T}}A)^{-1} - (B^{\mathrm{T}}B)^{-1}\|_2\|A\|_2 + \|(B^{\mathrm{T}}B)^{-1}\|_2\|\Delta\|_2\\ &\le \|(A^{\mathrm{T}}A)^{-1} - (B^{\mathrm{T}}B)^{-1}\|_2\|A\|_2 + \frac{\|\Delta\|_2}{\sigma_{\min}(B)^2}.\end{aligned}$$

We finally bound $\|(A^{\mathrm{T}}A)^{-1} - (B^{\mathrm{T}}B)^{-1}\|_2$.

$$\begin{aligned}\|(A^{\mathrm{T}}A)^{-1} - (B^{\mathrm{T}}B)^{-1}\|_2 &\le \frac{1}{\sigma_{\min}(A^{\mathrm{T}}A)}\|(A^{\mathrm{T}}A)((A^{\mathrm{T}}A)^{-1} - (B^{\mathrm{T}}B)^{-1})\|_2\\ &\le \frac{1}{\sigma_{\min}(A^{\mathrm{T}}A)}\|I - (A^{\mathrm{T}}A)(B^{\mathrm{T}}B)^{-1}\|_2\\ &\le \frac{1}{\sigma_{\min}(A^{\mathrm{T}}A)}\|I - (B^{\mathrm{T}}B + \Delta^{\mathrm{T}}B + B^{\mathrm{T}}\Delta + \Delta^{\mathrm{T}}\Delta)(B^{\mathrm{T}}B)^{-1}\|_2\\ &\le \frac{1}{\sigma_{\min}(A^{\mathrm{T}}A)}\|I - I - (\Delta^{\mathrm{T}}B + B^{\mathrm{T}}\Delta + \Delta^{\mathrm{T}}\Delta)(B^{\mathrm{T}}B)^{-1}\|_2\\ &\le \frac{2\|\Delta\|_2\|B\|_2 + \|\Delta\|_2^2}{\sigma_{\min}(A^{\mathrm{T}}A)\sigma_{\min}(B^{\mathrm{T}}B)}.\end{aligned}$$

We therefore obtain

$$\|AA^+ - BB^+\|_2 \le \|A\|_2^2\|(A^{\mathrm{T}}A)^{-1} - (B^{\mathrm{T}}B)^{-1}\|_2 + \frac{\|\Delta\|_2\|A\|_2}{\sigma_{\min}(B^{\mathrm{T}}B)} + \frac{\|\Delta\|_2}{\sigma_{\min}(B)}$$

$$\leq \frac{\|A\|_2^2}{\sigma_{\min}(A^{\mathrm{T}}A)} \frac{2\|\Delta\|_2\|B\|_2 + \|\Delta\|_2^2}{\sigma_{\min}(B^{\mathrm{T}}B)} + \frac{\|\Delta\|_2\|A\|_2}{\sigma_{\min}(B^{\mathrm{T}}B)} + \frac{\|\Delta\|_2}{\sigma_{\min}(B)}.$$

As $\|A - B\|_2 \leq \delta\|A\|_2$, we get that $(1-\delta)\|A\|_2 \leq \|B\|_2 \leq (1+\delta)\|A\|_2$. We also have that $\sigma_{\min}(B) \geq \sigma_{\min}(A) - \|A-B\|_2 \geq \|A\|_2/\kappa(A) - \delta\|A\|_2 \geq \|A\|_2/2\kappa(A) = \sigma_{\min}(A)/2$ if $\delta < 1/2\kappa(A)$. We can therefore conclude that $\|AA^+ - BB^+\|_2 \leq 20\delta\kappa(A)^4$. □

The condition that δ must be less than $1/2\kappa(A)$ in the above lemma makes sense as otherwise $20\delta\kappa(A)^4 \geq 10\kappa(A)^3 \geq 10$, which is a trivial upper bound on the norm.

Now we construct a matrix Apx that has its columns spanned by K' and is close to the matrix $p(M)G$. Using the bound on the condition number of the matrix $p(M)G$, we can conclude that the projection matrices onto the column spans of Apx and $p(M)G$, respectively, are close.

Recall $p(x)$ from (B.1). For q odd, the Chebyshev polynomial of degree q contains only odd degree monomials. So we have $T(x) = T_q x^q + T_{q-2}x^{q-2} + \ldots + T_1 x$ and therefore, the polynomial $p(x) = \frac{(1+\gamma)\alpha}{T(1+\gamma)}\left(\frac{T_q}{\alpha^q}x^q + \frac{T_{q-2}}{\alpha^{q-2}}x^{q-2} + \cdots + \frac{T_1}{\alpha_1}x\right)$, which implies

$$p(M)G = \frac{(1+\gamma)\alpha}{T(1+\gamma)}\left(\frac{T_q}{\alpha^q}(MM^{\mathrm{T}})^{(q-1)/2}MG + \cdots + \frac{T_1}{\alpha_1}MG\right).$$

We now define

$$\mathrm{Apx} = \frac{(1+\gamma)\alpha}{T(1+\gamma)}\left(\frac{T_q}{\alpha^q}(MM^{\mathrm{T}})^{\circ(q-1)/2}M \circ G + \cdots + \frac{T_1}{\alpha_1}M \circ G\right). \tag{B.3}$$

Clearly, the matrix Apx is spanned by the columns of the matrix K'. Using Lemma B.3.1 and properties of Gaussian matrices, the following lemma bounds $\|\mathrm{Apx} - p(M)G\|_2$.

Lemma B.3.8. *For the matrices $p(M)G$ and* Apx *defined above, we have with probability* $\geq 3/5$

$$\|p(M)G - \mathrm{Apx}\|_2 \leq \|p(M)G - \mathrm{Apx}\|_{\mathrm{F}} \leq 64C\varepsilon_\circ k^{3/2}(3\sqrt{2}\kappa)^q\|p(M)G\|_2.$$

Proof. By the triangle inequality,

$$\begin{aligned}
&\|p(M)G - \mathrm{Apx}\|_{\mathrm{F}} \\
&\leq \frac{(1+\gamma)\alpha}{T(1+\gamma)} \sum_{\text{odd } i \leq q} \frac{|T_i|}{\alpha^i}\|(MM^{\mathrm{T}})^{(i-1)/2}MG - (MM^{\mathrm{T}})^{\circ(i-1)/2}M \circ G\|_{\mathrm{F}} \\
&\leq \frac{(1+\gamma)\alpha}{T(1+\gamma)} \sum_{\text{odd } i \leq q} \frac{|T_i|}{\alpha^i}E_{i,G}
\end{aligned}$$

$$\leq \frac{(1+\gamma)\alpha}{T(1+\gamma)} \sum_{\text{odd } i \leq q} \frac{|T_i|}{\alpha^i} 8\varepsilon_\circ (2^{i/2}\|M\|_2^i \|G\|_{\mathrm{F}}) \qquad \text{(Lemma B.3.1)}$$

$$\leq \frac{(1+\gamma)\sigma_{k+1}(M)}{T(1+\gamma)} 8\varepsilon_\circ \|G\|_{\mathrm{F}} \sum_{\text{odd } i \leq q} |T_i|(\sqrt{2}\kappa)^i \qquad (\alpha = \sigma_{k+1}(M))$$

$$\leq \frac{(1+\gamma)\sigma_{k+1}(M)}{T(1+\gamma)} 8\varepsilon_\circ \|G\|_{\mathrm{F}} (3\sqrt{2}\kappa)^q \qquad (\sum_i |T_i| \leq 3^q)$$

$$\leq \|p(M)\|_2 8\varepsilon_\circ \|G\|_{\mathrm{F}} (3\sqrt{2}\kappa)^q. \qquad \text{(Equation B.2)}$$

We also condition on the following events both of which hold simultaneously with probability $\geq 4/5$.

- $\|G\|_{\mathrm{F}} \leq 4\sqrt{dk}$, and
- $\|p(M)G\|_2 \geq (1/C)\|p(M)\|_2(\sqrt{d} - \sqrt{k-1}) \geq (1/2C)\|p(M)\|_2\sqrt{d}.$

Thus, with probability $\geq 4/5$, if $d \geq 4k$,

$$\|p(M)G - \mathrm{Apx}\|_{\mathrm{F}} \leq \|p(M)\|_2(32\varepsilon_\circ)\sqrt{dk}(3\sqrt{2}\kappa)^q \leq 64C\varepsilon_\circ\sqrt{k}(3\sqrt{2}\kappa)^q\|p(M)G\|_2.$$

If $k \leq d \leq 4k$, then $\|p(M)G\|_2 \geq (1/C)\|p(M)\|_2(\sqrt{d} - \sqrt{k-1}) \geq (1/2C)\|p(M)\|_2(1/\sqrt{k})$ and $\|p(M)G - \mathrm{Apx}\|_{\mathrm{F}} \leq 64C\varepsilon_\circ k^{3/2}(3\sqrt{2}\kappa)^q\|p(M)G\|_2.$ □

Let $Y_1 \in \mathbb{R}^{n\times k}$ be an orthonormal basis for the column span of $p(M)G$ and $Y \in \mathbb{R}^{n\times k}$ be an orthonormal basis for the matrix Apx. We now have from Lemmas B.3.7 and B.3.8 that

$$\begin{aligned}\|YY^{\mathrm{T}} - Y_1Y_1^{\mathrm{T}}\|_2 &\leq O(\varepsilon_\circ k^{3/2}(3\sqrt{2}\kappa)^q\kappa(p(M)G)^4) = O(\varepsilon_\circ k^{3/2}(3\sqrt{2}\kappa)^q(k^2d^2 3^{4q}\kappa^{4q}))\\ &= \varepsilon_\circ C^q k^4 d^2 \kappa^{5q}\end{aligned}$$

for some constant C. Let $\delta \coloneqq \varepsilon_\circ C^q k^4 d^2\kappa^{5q}$. Hence,

$$\|Y_1Y_1^{\mathrm{T}} - YY^{\mathrm{T}}\|_2 \leq \delta. \tag{B.4}$$

For $l \leq k$ such that $\sigma_l(M) \geq (1+\varepsilon)\sigma_{k+1}(M)$, let $\mathscr{E}_l = \|[M]_l\|_{\mathrm{F}}^2 - \|Y_1Y_1^{\mathrm{T}}[M]_l\|_{\mathrm{F}}^2$ and $\mathscr{E}_l' = \|[M]_l\|_{\mathrm{F}}^2 - \|YY^{\mathrm{T}}[M]_l\|_{\mathrm{F}}^2$. Musco and Musco [MM15, Equation 7] show that

$$\mathscr{E}_l = \|[M]_l\|_{\mathrm{F}}^2 - \|Y_1Y_1^{\mathrm{T}}[M]_l\|_{\mathrm{F}}^2 \leq (\varepsilon/2)\sigma_{k+1}(M)^2.$$

Bounding $\mathscr{E}_l$ is one of the important steps in the analysis of [MM15]. We obtain a similar bound on $\mathscr{E}_l'$. We further show that if $M_{K',l}$ is the best rank l Frobenius norm approximation of M in colspan(K'), then $\|[M]_l\|_{\mathrm{F}}^2 - \|M_{K',l}\|_{\mathrm{F}}^2 \leq (3\varepsilon/4)\sigma_{k+1}(M)^2$, showing that there is a very good rank-l approximation for M in colspan(K'). We have the following lemma.

Lemma B.3.9. *Given a matrix A and a parameter k, let Y_1 be an orthonormal basis for a k dimensional subspace such that $\mathscr{E}_l = \|[M]_l\|_\mathrm{F}^2 - \|Y_1 Y_1^\mathrm{T}[M]_l\|_\mathrm{F}^2 \le (\varepsilon/2)\sigma_{k+1}^2$ for all $l \le k$ satisfying $\sigma_l(M) \ge (1+\varepsilon)\sigma_{k+1}(M)$. If Y is an orthonormal basis for another k dimensional subspace for which $\|YY^\mathrm{T} - Y_1 Y_1^\mathrm{T}\|_2 \le \varepsilon/(16\kappa^2\sqrt{k})$, where $\kappa = \sigma_1(M)/\sigma_{k+1}(M)$, then for all such l,*

$$\mathscr{E}_l' = \|[M]_l\|_\mathrm{F}^2 - \|YY^\mathrm{T}[M]_l\|_\mathrm{F}^2 \le (3\varepsilon/4)\sigma_{k+1}^2.$$

There also exists a matrix Y^l with l orthonormal columns with $\mathrm{colspan}(Y^l) \subseteq \mathrm{colspan}(K')$ such that $\|[M]_l\|_\mathrm{F}^2 - \|Y^l(Y^l)^\mathrm{T}M\|_\mathrm{F}^2 \le (3\varepsilon/4)\sigma_{k+1}^2$.

Proof. For any $1 > \varepsilon_\mathrm{s} > 0$

$$\begin{aligned}\|Y_1Y_1^\mathrm{T}[M]_l\|_\mathrm{F}^2 &\le (1+\varepsilon_\mathrm{s})\|YY^\mathrm{T}[M]_l\|_\mathrm{F}^2 + (1+\frac{1}{\varepsilon_\mathrm{s}})\|(YY^\mathrm{T} - Y_1Y_1^\mathrm{T})[M]_l\|_\mathrm{F}^2 \\ &\le (1+\varepsilon_\mathrm{s})\|YY^\mathrm{T}[M]_l\|_\mathrm{F}^2 + (2/\varepsilon_\mathrm{s})2k\delta^2\sigma_1(M)^2.\end{aligned}$$

The last inequality follows from the fact that $YY^\mathrm{T} - Y_1Y_1^\mathrm{T}$ has rank at most $2k$. Therefore,

$$\|YY^\mathrm{T}[M]_l\|_\mathrm{F}^2 \ge \frac{1}{1+\varepsilon_\mathrm{s}}\|Y_1Y_1^\mathrm{T}[M]_l\|_\mathrm{F}^2 - \frac{4k\sigma_1(M)^2}{\varepsilon_\mathrm{s}}\delta^2$$

which implies that

$$\begin{aligned}\mathscr{E}_l' &= \|[M]_l\|_\mathrm{F}^2 - \|YY^\mathrm{T}[M]_l\|_\mathrm{F}^2 \\ &\le \|[M]_l\|_\mathrm{F}^2 - \frac{1}{1+\varepsilon_\mathrm{s}}\|Y_1Y_1^\mathrm{T}[M]_l\|_\mathrm{F}^2 + \frac{4k\sigma_1(M)^2}{\varepsilon_\mathrm{s}}\delta^2 \\ &\le \frac{1}{1+\varepsilon_\mathrm{s}}(\|[M]_l\|_\mathrm{F}^2 - \|Y_1Y_1^\mathrm{T}[M]_l\|_\mathrm{F}^2) + \varepsilon_\mathrm{s}\|[M]_l\|_\mathrm{F}^2 + \frac{4k\sigma_1(M)^2}{\varepsilon_\mathrm{s}}\delta^2 \\ &\le \frac{1}{1+\varepsilon_\mathrm{s}}\frac{\varepsilon}{2}\sigma_{k+1}(M)^2 + \varepsilon_\mathrm{s}k\sigma_1(M)^2 + \frac{4k\sigma_1(M)^2}{\varepsilon_\mathrm{s}}\delta^2.\end{aligned}$$

Picking $\varepsilon_\mathrm{s} = \varepsilon/(8k\kappa^2)$ and if $\delta \le \varepsilon/(16\kappa^2\sqrt{k})$, we obtain that

$$\mathscr{E}_l' = \|[M]_l\|_\mathrm{F}^2 - \|YY^\mathrm{T}[M]_l\|_\mathrm{F}^2 \le \frac{3\varepsilon}{4}\sigma_{k+1}^2.$$

Recall here that $\kappa = \sigma_1(M)/\sigma_{k+1}(M)$. The matrix $YY^\mathrm{T}[M]_l$ is a rank l approximation for matrix M inside the column span of Y and hence in the column span of K'. Let Y^l be a rank l matrix that forms a basis for the best rank l approximation of M inside the column space of K' i.e.,

$$\min_{\text{rank-}l\ B:\mathrm{colspan}(B)\subseteq\mathrm{colspan}(K')} \|M - B\|_\mathrm{F}^2 = \|M - Y^lY^lM\|_\mathrm{F}^2.$$

From Lemma 6.2.3, note that if $\bar{U}\bar{\Sigma}^2\bar{V}^{\mathrm{T}}$ is the singular value decomposition of the matrix $(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'$ (recall Q' denotes an orthonormal basis for the matrix K'), then $Y^l = Q'\bar{U}_l$ where $\bar{U}_l$ denotes the first l columns of the matrix $\bar{U}$. By the optimality of Y^l, $\|M - Y^l(Y^l)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le \|M - YY^{\mathrm{T}}[M]_l\|_{\mathrm{F}}^2$ which implies that $\|YY^{\mathrm{T}}[M]_l\|_{\mathrm{F}}^2 \le \|Y^l(Y^l)^{\mathrm{T}}M\|_{\mathrm{F}}^2$. Thus, $\|[M]_l\|_{\mathrm{F}}^2 - \|Y^l(Y^l)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le \|[M]_l\|_{\mathrm{F}}^2 - \|YY^{\mathrm{T}}[M]_l\|_{\mathrm{F}}^2 = \mathscr{E}_l' \le (3\varepsilon/4)\sigma_{k+1}^2$. □

The proof also shows that if $\bar{U}\bar{\Sigma}^2\bar{U}^{\mathrm{T}}$ is the singular value decomposition of the positive semi-definite matrix $(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'$, then $Y^l = Q'\bar{U}_l$ where $\bar{U}_l$ denotes the matrix that contains the first l columns of $\bar{U}$. Let $m \le k$ be the largest integer for which $\sigma_m(M) \ge (1+\varepsilon)\sigma_{k+1}(M)$. From the above lemma, the matrix Y^m satisfies $\|[M]_l\|_{\mathrm{F}}^2 - \|Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le (3\varepsilon/4)\sigma_{k+1}(M)^2$. We later show that this implies $\|M - Y^m(Y^m)^{\mathrm{T}}M\|_2 \le (1+3\varepsilon/2)\sigma_{k+1}(M)$. Unfortunately, we cannot compute the matrix $(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'$ exactly as we only have access to an oracle that computes vector products with matrices M, M^{T} approximately. Nevertheless, we show that we can compute a matrix $\hat{Y}^m$ based on an approximation to the matrix $(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'$ and it still satisfies the desired guarantees approximately.

First we have the following lemma that shows if a subspace Y^m is a good approximation for Frobenius norm low rank approximation of M in m dimensions, then the subspace Y^m is also a good subspace for spectral norm rank-k approximation of matrix M. It also shows that even if $\hat{Y}^m$ only approximately satisfies the properties of Y^m, the matrix $\hat{Y}^m$ spans a good low rank approximation for M.

Lemma B.3.10. *Given an arbitrary matrix M, if an orthonormal basis Y^m to an m-dimensional subspace, where $m \le k$ is the largest integer such that $\sigma_m(M) \ge (1+\varepsilon)\sigma_{k+1}(M)$, satisfies*

$$\|[M]_m\|_{\mathrm{F}}^2 - \|Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le \varepsilon\sigma_{k+1}(M)^2,$$

then $\|M - Y^m(Y^m)^{\mathrm{T}}M\|_2 \le (1+2\varepsilon)\sigma_{k+1}(M)$. Additionally, if $\hat{Y}^m$ is a matrix with m orthonormal columns such that

$$\|M - \hat{Y}^m(\hat{Y}^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le \|M - Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 + \delta,$$

then $\|M - \hat{Y}^m(\hat{Y}^m)^{\mathrm{T}}M\|_2 \le (1+2\varepsilon)\sigma_{k+1}(M) + \sqrt{\delta}$.

Proof. As $\|[M]_m\|_{\mathrm{F}}^2 - \|Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 = \|M\|_{\mathrm{F}}^2 - \|M - [M]_m\|_{\mathrm{F}}^2 - \|Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 = \|M - Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 - \|M - [M]_m\|_{\mathrm{F}}^2$, we obtain that

$$\|M - Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le \|M - [M]_m\|_{\mathrm{F}}^2 + \varepsilon\sigma_{k+1}(M)^2.$$

As an additive error in Frobenius norm translates to additive error in spectral norm for the above case (see Theorem 3.2 from [Gu15]), we obtain

$$\|M - Y^m(Y^m)^{\mathrm{T}}M\|_2^2 \le \|M - [M]_m\|_2^2 + \varepsilon\sigma_{k+1}(M)^2 \le \sigma_{m+1}(M)^2 + \varepsilon\sigma_{k+1}(M)^2$$
$$\le (1+4\varepsilon)\sigma_{k+1}(M)^2.$$

Thus, $\|M - Y^m(Y^m)^{\mathrm{T}}M\|_2 \le (1+2\varepsilon)\sigma_{k+1}(M)$. Similarly, we have that

$$\|M - \hat{Y}^m(\hat{Y}^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le \|M - [M]_m\|_{\mathrm{F}}^2 + \varepsilon\sigma_{k+1}(M)^2 + \delta$$

which implies that

$$\|M - \hat{Y}^m(\hat{Y}^m)^{\mathrm{T}}M\|_2^2 \le \|M - [M]_m\|_2^2 + \varepsilon\sigma_{k+1}(M)^2 + \delta \le (1+4\varepsilon)\sigma_{k+1}(M)^2 + \delta$$

which shows $\|M - \hat{Y}^m(\hat{Y}^m)^{\mathrm{T}}M\|_2 \le (1+2\varepsilon)\sigma_{k+1}(M) + \sqrt{\delta}$. □

The above lemma shows that we need only compute a matrix $\hat{Y}^m$ such that $\|M-\hat{Y}^m(\hat{Y}^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \approx \|M - Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2$.

We show that using an approximation to matrix $(Q')^{\mathrm{T}}MM^{\mathrm{T}}(Q')^{\mathrm{T}}$ we can compute such a matrix $\hat{Y}^m$ which shows that $\|M - \hat{Y}^m(\hat{Y}^m)^{\mathrm{T}}M\|_2 \le (1+O(\varepsilon))\sigma_{k+1}(M)$. As the value of $m \le k$ is not known, we further show that we can compute a matrix $\hat{Y}^k$ with k orthonormal columns such that $\mathrm{colspan}(M) \supseteq \mathrm{colspan}(K') \supseteq \mathrm{colspan}(\hat{Y}^k) \supseteq \mathrm{colspan}(\hat{Y}^m)$. Therefore, we can conclude that $\|M - \hat{Y}^k(\hat{Y}^k)^{\mathrm{T}}M\|_2 \le \|M - \hat{Y}^k(\hat{Y}^k)^{\mathrm{T}}M\|_2 \le (1+O(\varepsilon))\sigma_{k+1}(M)$. We thus have our final result for low rank approximation.

B.3.3 Proof of Theorem 6.5.1

Computing top k singular vectors of the matrix $(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'$ We now show that if $\hat{Y}^m$ are the top m singular vectors of the matrix $(Q')^{\mathrm{T}}((MM^{\mathrm{T}}) \circ Q')$, then

$$\|M - \hat{Y}^m(\hat{Y}^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \approx \|M - Y^m(Y^m)^{\mathrm{T}}M\|_{\mathrm{F}}^2.$$

Lemma B.3.11. *If Z_m are the top m orthonormal eigenvectors of the matrix MM^{T}, then for any matrix Y with m orthonormal columns,*

$$\mathrm{tr}(Z_m^{\mathrm{T}}MM^{\mathrm{T}}Z_m) \ge \mathrm{tr}(Y^{\mathrm{T}}MM^{\mathrm{T}}Y).$$

Proof. We have $\mathrm{tr}(Z_m^{\mathrm{T}}MM^{\mathrm{T}}Z_m) = \|Z_mZ_m^{\mathrm{T}}M\|_{\mathrm{F}}^2$ and $\mathrm{tr}(Y^{\mathrm{T}}MM^{\mathrm{T}}Y) = \|YY^{\mathrm{T}}M\|_{\mathrm{F}}^2$. We are given that Z_m are the top m eigenvectors of the matrix MM^{T} and therefore Z_m are the top m singular vectors of the matrix M. Therefore, for any matrix Y with m orthonormal columns, we have that $\|Z_mZ_m^{\mathrm{T}}M\|_{\mathrm{F}}^2 \ge \|YY^{\mathrm{T}}M\|_{\mathrm{F}}^2$ and therefore that $\mathrm{tr}(Z_m^{\mathrm{T}}MM^{\mathrm{T}}Z_m) \ge \mathrm{tr}(Y^{\mathrm{T}}MM^{\mathrm{T}}Y)$. □

Lemma B.3.12. *Let M be a matrix and Q be an orthonormal basis for an arbitrary r dimensional space. Let B be a positive semi-definite matrix such that $B - Q^{\mathrm{T}}MM^{\mathrm{T}}Q = \Delta$. Let Z be a matrix whose columns are the top k eigenvectors of the matrix B. Then if Z_m denotes the matrix with first m columns of Z for $m = 1, \dots, k$ we have*

$$\|M - (QZ_m)(QZ_m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le \|M - Q(Q^{\mathrm{T}}M)_m\|_{\mathrm{F}}^2 + 2m\|\Delta\|_{\mathrm{F}}.$$

Proof. Let Z^* be the matrix whose columns are the top k eigenvectors of the matrix $Q^{\mathrm{T}}MM^{\mathrm{T}}Q$ and Z_m^* be the first m columns of Z^*. Thus, $Q(Q^{\mathrm{T}}M)_m = Q(Z_m^*(Z_m^*)^{\mathrm{T}}Q^{\mathrm{T}}M) = (QZ_m^*)(QZ_m^*)^{\mathrm{T}}M$. Now,

$$\begin{aligned}
\|(QZ_m)(QZ_m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 &= \|(QZ_m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \\
&= \mathrm{tr}(Z_m^{\mathrm{T}}Q^{\mathrm{T}}MM^{\mathrm{T}}QZ_m) \\
&= \mathrm{tr}(Z_m^{\mathrm{T}}(Q^{\mathrm{T}}MM^{\mathrm{T}}Q + \Delta)Z_m) - \mathrm{tr}(Z_m^{\mathrm{T}}\Delta Z_m) \\
&= \mathrm{tr}(Z_m^{\mathrm{T}}BZ_m) - \mathrm{tr}(Z_m^{\mathrm{T}}\Delta Z_m) \\
&\ge \mathrm{tr}((Z_m^*)^{\mathrm{T}}BZ_m^*) - m\|\Delta\|_{\mathrm{F}} \\
&\quad (\text{Since } \mathrm{tr}(Z_m^{\mathrm{T}}\Delta Z_m) = \mathrm{tr}(\Delta Z_m Z_m^{\mathrm{T}}) \le \|\Delta\|_{\mathrm{F}}\|Z_m Z_m^{\mathrm{T}}\|_{\mathrm{F}} \le \|\Delta\|_{\mathrm{F}} \cdot m) \\
&= \mathrm{tr}((Z_m^*)^{\mathrm{T}}(Q^{\mathrm{T}}MM^{\mathrm{T}}Q)Z_m^*) - \mathrm{tr}((Z_m^*)^{\mathrm{T}}\Delta Z_m^*) - m\|\Delta\|_{\mathrm{F}} \\
&= \mathrm{tr}(QZ_m^*(Z_m^*)^{\mathrm{T}}Q^{\mathrm{T}}MM^{\mathrm{T}}QZ_m^*(Z_m^*)^{\mathrm{T}}Q^{\mathrm{T}}) - \mathrm{tr}((Z_m^*)^{\mathrm{T}}\Delta Z_m^*) - m\|\Delta\|_{\mathrm{F}} \\
&\ge \|(QZ_m^*)(QZ_m^*)^{\mathrm{T}}M\|_{\mathrm{F}}^2 - 2m\|\Delta\|_{\mathrm{F}}.
\end{aligned}$$

Thus,

$$\|M - (QZ_m)(QZ_m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 \le \|M - (QZ_m^*)(QZ_m^*)^{\mathrm{T}}M\|_{\mathrm{F}}^2 + 2m\|\Delta\|_{\mathrm{F}},$$

which concludes the proof. □

Hence, if $\widetilde{\mathrm{Apx}}$ is a positive semi-definite matrix such that $\|\widetilde{\mathrm{Apx}} - (Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'\|_{\mathrm{F}}$ is small and if Z_m denotes the top m singular vectors of the matrix $\widetilde{\mathrm{Apx}}$, we can conclude by Lemma B.3.10 that $\|M - (Q'Z_m)(Q'Z_m)^{\mathrm{T}}M\|_2$ is close to $\sigma_{k+1}(M)$.

We now show that we can compute such a matrix $\widetilde{\mathrm{Apx}}$. Let $\Xi = (Q')^{\mathrm{T}}((MM^{\mathrm{T}}) \circ Q')$ (recall that $\circ$ denotes matrix multiplication using the noisy oracle). Let $\widetilde{\mathrm{Apx}} = \mathrm{psd}((\Xi + \Xi^{\mathrm{T}})/2)$. Then the following lemma shows that $\widetilde{Apx}$ is close to $(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'$.

Lemma B.3.13. *Given matrices $M \in \mathbb{R}^{n\times d}$ and $Q' \in \mathbb{R}^{n\times t}$ where Q' is a matrix with t orthonormal columns, if for all vectors v, v', $\|M \circ v - Mv\|_2 \le \varepsilon_\circ\|M\|_2\|v\|_2$ and $\|M^{\mathrm{T}} \circ v' - M^{\mathrm{T}}v'\|_2 \le \varepsilon_\circ\|M\|_2\|v'\|_2$, and $\Xi \coloneqq (Q')^{\mathrm{T}}(MM^{\mathrm{T}}) \circ Q'$, then*

$$\|\mathrm{psd}((\Xi + \Xi^{\mathrm{T}})/2) - (Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'\|_{\mathrm{F}} \le (6\varepsilon_\circ\|M\|_2^2)\sqrt{t}.$$

Let $\widetilde{Apx} = \mathrm{psd}((\Xi + \Xi^{\mathrm{T}})/2)$. The matrix $\widetilde{\mathrm{Apx}}$ can be computed in time $O(2tT(\varepsilon_\circ) + t^3)$.

Proof. Let k_i be the i^{th} column of the matrix K' and $E_i = \|(Q')^{\mathrm{T}}(MM^{\mathrm{T}}) \circ k_i - (Q')^{\mathrm{T}}(MM^{\mathrm{T}})k_i\|_2$. Then

$$\begin{aligned}
E_i &= \|(Q')^{\mathrm{T}}(MM^{\mathrm{T}}) \circ k_i - (Q')^{\mathrm{T}}(MM^{\mathrm{T}})k_i\|_2 \\
&\leq \|(MM^{\mathrm{T}}) \circ k_i - (MM^{\mathrm{T}})k_i\|_2 \\
&= \|M \circ (M^{\mathrm{T}} \circ k_i) - M(M^{\mathrm{T}}k_i)\|_2 \\
&\leq \|M \circ (M^{\mathrm{T}} \circ k_i) - M(M^{\mathrm{T}} \circ k_i) + M(M^{\mathrm{T}} \circ k_i) - M(M^{\mathrm{T}}k_i)\|_2 \\
&\leq \|M \circ (M^{\mathrm{T}} \circ k_i) - M(M^{\mathrm{T}} \circ k_i)\|_2 + \|M(M^{\mathrm{T}} \circ k_i) - M(M^{\mathrm{T}}k_i)\|_2 \\
&\leq \varepsilon_\circ\|M\|_2\|M^{\mathrm{T}} \circ k_i\|_2 + \|M\|_2\varepsilon_\circ\|M\|_2\|k_i\|_2 \\
&\leq \varepsilon_\circ\|M\|_2(\|M^{\mathrm{T}}k_i\|_2 + \varepsilon_\circ\|M\|_2\|k_i\|_2) + \|M\|_2^2\varepsilon_\circ\|k_i\|_2 \\
&\leq 3\varepsilon_\circ\|M\|_2^2. && \text{(Since } \|k_i\|_2 = 1\text{)}
\end{aligned}$$

Thus $\|(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q' - \Xi\|_{\mathrm{F}}^2 = \sum_{i=1}^t \|(Q')^{\mathrm{T}}MM^{\mathrm{T}}k_i - (Q')^{\mathrm{T}}(MM^{\mathrm{T}}) \circ k_i\|_2^2 \leq (3\varepsilon_\circ\|M\|_2^2)^2 t$ which implies that $\|(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q' - \Xi\|_{\mathrm{F}} \leq (3\varepsilon_\circ\|M\|_2^2)\sqrt{t}$. Now as $(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'$ is a symmetric matrix, $\|(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q' - (\Xi+\Xi^{\mathrm{T}})/2\|_{\mathrm{F}} \leq (3\varepsilon_\circ\|M\|_2^2)\sqrt{t}$. As $(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q'$ is itself a positive semidefinite matrix,

$$\|\mathrm{psd}((\Xi+\Xi^{\mathrm{T}})/2) - (\Xi+\Xi^{\mathrm{T}})/2\|_{\mathrm{F}} \leq \|(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q' - (\Xi+\Xi^{\mathrm{T}})/2\|_{\mathrm{F}} \leq (3\varepsilon_\circ\|M\|_2^2)\sqrt{t}.$$

Finally, by the triangle inequality we obtain that $\|(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q' - \widetilde{Apx}\|_{\mathrm{F}} = \|(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q' - \mathrm{psd}((\Xi+\Xi^{\mathrm{T}})/2)\|_{\mathrm{F}} \leq 6\varepsilon_\circ\|M\|_2^2\sqrt{t}$. The time required to compute matrix Ξ is $2tT(\varepsilon_\circ) + nt^2$ and $\mathrm{psd}((\Xi+\Xi^{\mathrm{T}})/2)$ is $O(t^3)$. Thus, the matrix $\widetilde{\mathrm{Apx}}$ can be computed in time $O(2tT(\varepsilon_\circ) + t^3)$. □

Proof of Theorem 6.5.1. Let $q = O((1/\sqrt{\varepsilon})\log(d/\varepsilon))$. Algorithm 6.1 computes the Krylov subspace K' with

$$\varepsilon_\circ = \frac{\varepsilon}{16\kappa^{2+5q}k^5d^2C^q}$$

for an absolute constant C. Let Y_1 be an orthonormal basis for $p(M)G$ and Y be an orthonormal basis for the matrix Apx (defined in (B.3)). Then by (B.4) we have that $\|YY^{\mathrm{T}} - Y_1Y_1^{\mathrm{T}}\|_2 \leq \varepsilon/(16\kappa^2\sqrt{k})$. If $m \leq k$ is the largest integer such that $\sigma_m(M) \geq (1+\varepsilon)\sigma_{k+1}(M)$, by Lemma B.3.9, there exists a d dimensional subspace Y^m inside the column span of K' such that

$$\|[M]_m\|_{\mathrm{F}}^2 - \|Y^m(Y^m)^{\mathrm{T}}A\|_{\mathrm{F}}^2 \leq (3\varepsilon/4)\sigma_{k+1}^2.$$

If Ξ is now computed with $\varepsilon_o = \varepsilon^2/(48\kappa^2(\sqrt{qk})k)$, then by Lemma B.3.13,

$$\|(Q')^{\mathrm{T}}MM^{\mathrm{T}}Q' - \widetilde{\mathrm{Apx}}\|_{\mathrm{F}} \leq \frac{\varepsilon}{8k}\sigma_{k+1}^2.$$

Now if Z_k denotes the first k singular vectors of the matrix $\widetilde{Apx}$, and Z_m denotes the first m columns of Z_k, then by Lemma B.3.12, we get that

$$\begin{aligned}\|M - (Q'Z_m)(Q'Z_m)^{\mathrm{T}}M\|_{\mathrm{F}}^2 &\leq \|M - Q'((Q')^{\mathrm{T}}M)_m\|_{\mathrm{F}}^2 + 2m(\frac{\varepsilon^2}{8k}\sigma_{k+1}^2) \\ &\leq \|M - Q'((Q')^{\mathrm{T}}M)_m\|_{\mathrm{F}}^2 + \frac{\varepsilon^2}{4}\sigma_{k+1}^2.\end{aligned}$$

Finally, by Lemma B.3.10, we obtain that

$$\|M - (Q'Z_m)(Q'Z_m)^{\mathrm{T}}M\|_2 \leq (1 + 3\varepsilon/2)\sigma_{k+1} + \sqrt{(\varepsilon^2/4)\sigma_{k+1}^2} \leq (1 + 2\varepsilon)\sigma_{k+1}.$$

Also, $\|M - (Q'Z_k)(Q'Z_k)^{\mathrm{T}}M\|_2 \leq \|M - (Q'Z_m)(Q'Z_m)^{\mathrm{T}}M\|_2 \leq (1 + 2\varepsilon)\sigma_{k+1}(M)$ since $Q'Z_k$ has orthonormal columns and $\mathrm{colspan}(Q'Z_k) \supseteq \mathrm{colspan}(Q'Z_m)$. Thus, in time

$$T\left(\frac{\varepsilon}{\kappa^{5q}k^5d^2C^q}\right)qk + T\left(\frac{\varepsilon^2}{48\kappa^2(\sqrt{qk})k}\right)qk,$$

Algorithm 6.1 computes a $1 + 2\varepsilon$ approximation. Scaling the value of ε gives us the result. If the approximations $M \circ v$ are spanned by the column space of M for all vectors v, then the columns of K' are spanned by the matrix M. Thus, the columns of Q' are also spanned by M, which implies that the columns of the matrix $Q'Z_m$ are spanned by M. □

B.4 Omitted Proofs in Section 6.6

B.4.1 Proof of Lemma 6.6.1

Proof. Define $Z \coloneqq U^T\widetilde{Z}$. We have

$$\begin{aligned}1 + \varepsilon &\geq \|AA^+B(\beta^2I - \Delta)^{-1/2} - \widetilde{Z}\widetilde{Z}^{\mathrm{T}}AA^+B(\beta^2I - \Delta)^{-1/2}\|_2 \\ &\geq \|UU^{\mathrm{T}}B(\beta^2I - \Delta)^{-1/2} - \widetilde{Z}\widetilde{Z}^{\mathrm{T}}UU^{\mathrm{T}}B(\beta^2I - \Delta)^{-1/2}\|_2 \\ &\geq \|UU^{\mathrm{T}}B(\beta^2I - \Delta)^{-1/2} - UU^{\mathrm{T}}\widetilde{Z}\widetilde{Z}^{\mathrm{T}}UU^{\mathrm{T}}B(\beta^2I - \Delta)^{-1/2}\|_2 \\ &= \|UU^{\mathrm{T}}B(\beta^2I - \Delta)^{-1/2} - UZZ^{\mathrm{T}}U^{\mathrm{T}}B(\beta^2I - \Delta)^{-1/2}\|_2 \\ &= \|U^{\mathrm{T}}B(\beta^2I - \Delta)^{-1/2} - ZZ^{\mathrm{T}}U^{\mathrm{T}}B(\beta^2I - \Delta)^{-1/2}\|_2\end{aligned}$$

which implies using Lemma 6.4.2 that $UZ = UU^{\mathrm{T}}\widetilde{Z} = AA^+\widetilde{Z}$ is a good space to project the columns of B onto, i.e.,

$$\|(AA^+\widetilde{Z})(AA^+\widetilde{Z})^+B - B\|_2 \leq (1 + \varepsilon)\beta.$$ □

B.4.2 Proof of Lemma 6.6.2

Polynomial Approximation of $(1-x)^{-1/2}$. We want to obtain a polynomial $p(x)$ such that $|p(x)-(1-x)^{-1/2}| \leq \delta$ in the interval $x \in [0, 1/(1+\varepsilon)]$. Consider the Taylor expansion of $(1-x)^{-1/2}$:

$$(1-x)^{-1/2} = \sum_{j=0}^{\infty} \frac{(2j)!}{2^{2j} j!^2} x^j.$$

The above series converges for all $|x| < 1$. Let $q(x)$ be the Taylor series up to T terms. Then for $1 > x \geq 0$, we have $0 \leq q(x) \leq (1-x)^{-1/2}$ and for $0 \leq x \leq 1/(1+\varepsilon)$

$$(1-x)^{-1/2} - q(x) = \sum_{j=T}^{\infty} \frac{(2j)!}{2^{2j} j!^2} x^j \leq \sum_{j=T}^{\infty} x^j = \frac{x^T}{1-x} \leq \frac{(1+\varepsilon)}{\varepsilon(1+\varepsilon)^T} = \frac{1}{\varepsilon(1+\varepsilon)^{T-1}}.$$

Thus, if $T-1 \geq 4\log(1/(\varepsilon\delta))/\varepsilon \geq \log(1/\varepsilon\delta)/\log(1+\varepsilon)$, we have $(1+\varepsilon)^{T-1} \geq 1/\varepsilon\delta$ which implies that

$$0 \leq (1-x)^{-1/2} - q(x) \leq \delta$$

for all $0 \leq x \leq 1/(1+\varepsilon)$. So, there is a degree $t = O(\log(1/\varepsilon\delta)/\varepsilon)$ polynomial that uniformly approximates $(1-x)^{-1/2}$ up to an error δ in the interval $[0, 1/(1+\varepsilon)]$. Now, we further approximate the degree t polynomial $q(x)$ with a degree $\widetilde{O}(\sqrt{t})$ polynomial.

First we have the following theorem.

Theorem B.4.1 (Theorem 3.3 of [SV14]). *For any positive integers s and d, there is a degree d polynomial $p_{s,d}(x)$ that satisfies*

$$\sup_{x\in[-1,1]} |p_{s,d}(x) - x^s| \leq 2e^{-d^2/2s}.$$

Further, this polynomial $p_{s,d}$ is defined as follows

$$p_{s,d}(x) = \mathbf{E}_{Y_1,\ldots,Y_s}[T_{|D|}(x)\mathbf{1}[|D| \leq d]]$$

where $Y_1, \ldots, Y_s$ are independent Rademacher random variables, $D = \sum_{i=1}^{s} Y_i$ and $\mathbf{1}$ denotes the indicator function.

Clearly the polynomial $p_{s,d}$ is defined as a weighted linear combination of Chebyshev polynomials of various degrees at most d. With $d = \sqrt{2s\log(1/\delta)}$, we have that

$$\sup_{x\in[-1,1]} |p_{s,d}(x) - x^s| \leq 2e^{-\log(1/\delta)} \leq 2\delta.$$

Thus, given an arbitrary degree t polynomial $q(x) = \sum_{i=0}^{t} q_i x^i$, where $q_0, \ldots, q_t$ are the coefficients of the polynomial, then the degree d polynomial $r(x) = \sum_{i=0}^{t} q_i p_{i,d}(x)$ with $d = \sqrt{2t\log(1/\delta)}$

satisfies

$$\begin{aligned}\sup_{x\in[-1,1]} |q(x)-r(x)| &= \sup_{x\in[-1,1]} |\sum_{i=0}^{t} q_i x^i - \sum_{i=0}^{t} q_i p_{i,d}(x)| \\ &\leq \sup_{x\in[-1,1]} \sum_{i=0}^{t} |q_i||x^i - p_{i,d}(x)| \\ &\leq \sup_{x\in[-1,1]} \sum_{i=0}^{t} |q_i| 2\delta \\ &= 2\|q\|_1 \delta.\end{aligned}$$

We now bound $\|r\|_1$. We have

$$\begin{aligned}\|r\|_1 = \|\sum_i q_i p_{i,d}(x)\| &\leq \sum_i |q_i| \|p_{i,d}(x)\|_1 \\ &= \sum_i |q_i| \| \mathbf{E}_{Y_1,\dots,Y_s}[T_{|D|}(x)\mathbf{1}[|D| \leq d]]\|_1 \\ &\leq \sum_i |q_i| \mathbf{E}_{Y_1,\dots,Y_s}[\|T_{|D|}(x)\mathbf{1}[|D|\leq d]]\|_1] \\ &\leq \sum_i |q_i| \frac{1}{2}(1+\sqrt{2})^d = \frac{1}{2}(1+\sqrt{2})^d \|q\|_1.\end{aligned}$$

Here we use the fact that $\|\cdot\|_1$ is convex over polynomials and that the sum of absolute values of coefficients of a Chebyshev polynomial of degree d is bounded by $(1+\sqrt{2})^d$. Thus, we have the following lemma.

Lemma B.4.2. *Given any polynomial $q(x)$ of degree t, there exists a polynomial $r(x)$ of degree $d = \sqrt{2t \log(2\|q\|_1/\delta)}$ such that*

$$\sup_{x\in[-1,1]} |q(x) - r(x)| \leq \delta$$

and $\|r\|_1 \leq (1+\sqrt{2})^d \|q\|_1$.

We already saw that the polynomial $q(x) = \sum_{j=0}^{t} \frac{(2j)!}{2^{2j} j!^2} x^j$ satisfies $|q(x) - (1-x)^{-1/2}| \leq \delta$ for $x \in [0, 1/(1+\varepsilon)]$ if $t = O(\log(1/\varepsilon\delta)/\varepsilon)$. We also have $\|q\|_1 = \sum_{j=0}^{t} |(2j)!/(2^{2j}(j!)^2)| \leq t+1$. Thus, by the above lemma, we can compute a polynomial $r(x)$ of degree $d = O(\sqrt{t\log(t/\delta)}) = O(\frac{1}{\sqrt{\varepsilon}} \log(1/\varepsilon\delta))$ such that

$$\sup_{x\in[0,1/(1+\varepsilon)]} |r(x) - (1-x)^{-1/2}| \leq \sup_{x\in[0,1/(1+\varepsilon)]} |(1-x)^{-1/2} - q(x)| + \sup_{x\in[-1,1]} |q(x) - r(x)| \leq 2\delta$$

and we also have $\|r\|_1 = O((1+\sqrt{2})^d t) = O((1+\sqrt{2})^{O(\sqrt{1/\varepsilon}\log(1/\varepsilon\delta))} \log(1/\varepsilon\delta)/\varepsilon)$. We summarize this in the following lemma.

Lemma B.4.3. *Given $\varepsilon, \delta > 0$, there exists a polynomial $r(x)$ of degree $O(\frac{1}{\sqrt{\varepsilon}}\log(1/\varepsilon\delta))$ and $\|r\|_1 = O((1+\sqrt{2})^{O(\sqrt{1/\varepsilon}\log(1/\varepsilon\delta))}\log(1/\varepsilon\delta)/\varepsilon)$ such that*

$$\sup_{x\in[0,1/(1+\varepsilon)]} |r(x) - (1-x)^{-1/2}| \le \delta.$$

Lemma B.4.4 (Matrix Approximation Lemma)**.** *If $A \in \mathbb{R}^{n\times n}$ is a positive semidefinite matrix with $\lambda_{\max}(A) < 1$ and if $r(x)$ is a polynomial such that*

$$\sup_{x\in[0,\lambda_{\max}(A)]} |r(x) - (1-x)^{-1/2}| \le \delta,$$

then $\|r(A) - (I-A)^{-1/2}\|_2 \le \delta$.

Proof. Let $A = VDV^{\mathrm{T}}$ be the eigenvalue decomposition of D with $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ where $\lambda_{\max} = \lambda_1 \ge \ldots \ge \lambda_n \ge 0$. Then $(I-A)^{-1/2} = V(I-D)^{-1/2}V^{\mathrm{T}}$ and $r(A) = Vr(D)V^{\mathrm{T}}$. Therefore,

$$\begin{aligned}\|r(A) - (I-A)^{-1/2}\|_2 &= \|V((I-D)^{-1/2} - r(D))V^{\mathrm{T}}\|_2 \\ &= \|(I-D)^{-1/2} - r(D)\|_2 \\ &= \max_i |(1-\lambda_i)^{-1/2} - r(\lambda_i)| \\ &\le \sup_{x\in[0,\lambda_{\max}(A)]} |(1-x)^{-1/2} - r(x)| \le \delta.\end{aligned}$$

Here we use the fact that $0 \le \lambda_1, \ldots, \lambda_n \le \lambda_{\max}(A)$. □

As Δ is a positive semidefinite matrix such that $\beta^2 \ge (1+\varepsilon)\|\Delta\|_2$, then $\|\Delta/\beta^2\|_2 \le 1/(1+\varepsilon)$ and hence we can compute a polynomial $r(x)$ of degree $O(\frac{1}{\sqrt{\varepsilon}}\log(1/\varepsilon\delta))$ such that

$$\|r(\Delta/\beta^2) - (I - \Delta/\beta^2)^{-1/2}\|_2 \le \delta.$$

Modified Problem. Instead of considering the matrix $\mathcal{M} = AA^+B(\beta^2 I - \Delta)^{-1/2}$ for low rank approximation, we consider the matrix $\mathcal{M}' = AA^+BM/\beta$ for $M = r(\Delta/\beta^2)$, where $r(x)$ is a low degree polynomial, and argue that a $(1+\varepsilon)$-approximate LRA solution for the matrix $\mathcal{M}'$ is a $1+2\varepsilon$ approximation for the LRA problem on matrix $\mathcal{M}$.

Proof of Lemma 6.6.2. Recall $\Delta = B^{\mathrm{T}}(I - AA^+)B$, and therefore $\|\Delta\|_2 = \|(I-AA^+)B\|_2^2$. Given that $\beta \ge (1+\varepsilon)\max(\|(I-AA^+)B\|_2, \sigma_{k+1}(B))$, we have $\beta^2 \ge (1+\varepsilon)^2\|\Delta\|_2 \ge (1+\varepsilon)\|\Delta\|_2$. Thus, $\|\Delta/\beta^2\|_2 \le 1/(1+\varepsilon)$.

As $\|\Delta/\beta^2\|_2 \le 1/(1+\varepsilon)$, we approximate $(I-\Delta/\beta^2)^{-1/2}$ with the matrix $M = r(\Delta/\beta^2)$ where $r(x) = \sum_{i=0}^{t} r_i x^i$ is a polynomial of degree $t = O(\frac{1}{\sqrt{\varepsilon}}\log(\frac{1}{\varepsilon\delta}))$ given by Lemma B.4.3. By Lemma B.4.4, the matrix $r(\Delta/\beta^2) = \sum_{i=0}^{t} r_i(\Delta/\beta^2)^i$ satisfies

$$\begin{aligned}\|(I - \Delta/\beta^2)^{-1/2} - M\|_2 &= \|(I - \Delta/\beta^2)^{-1/2} - r(\Delta/\beta^2)\|_2 \\ &= \|(I - \Delta/\beta^2)^{-1/2} - \sum_{i=0}^{t} r_i \left(\frac{\Delta}{\beta^2}\right)^i \|_2 \leq \delta.\end{aligned}$$

As $\|\Delta/\beta^2\|_2 \leq 1/(1+\varepsilon)$ and Δ/β^2 is a positive semidefinite matrix, we have $\sigma_{\max}(I - \Delta/\beta^2) \leq 1$ and $\sigma_{\min}(I - \Delta/\beta^2) \geq \varepsilon/(1+\varepsilon)$. Therefore, $\sigma_{\max}((I - \Delta/\beta^2)^{-1/2}) \leq \sqrt{(1+\varepsilon)/\varepsilon}$ and $\sigma_{\min}((I - \Delta/\beta^2)^{-1/2}) \geq 1$. By Weyl's inequality, we obtain that

$$\sigma_{\max}(M) \leq \sqrt{(1+\varepsilon)/\varepsilon} + \delta \quad \text{and} \quad \sigma_{\min}(M) \geq 1 - \delta.$$

By sub-multiplicativity, of the spectral norm

$$\begin{aligned}\|AA^+B(\beta^2 I - \Delta)^{-1/2} - \frac{AA^+BM}{\beta}\|_2 &\leq \frac{\|AA^+B\|_2}{\beta}\|(I - (\Delta/\beta^2))^{-1/2} - M\|_2 \\ &\leq \frac{\|AA^+B\|_2}{\beta}\delta.\end{aligned}$$

Using Weyl's inequality, we obtain

$$\sigma_{k+1}\left(\frac{AA^+BM}{\beta}\right) \leq \sigma_{k+1}(AA^+B(\beta^2 I - \Delta)^{-1/2}) + \frac{\|AA^+B\|_2}{\beta}\delta \leq 1 + \frac{\|AA^+B\|_2}{\beta}\delta. \tag{B.5}$$

The last inequality follows as there exists a rank k matrix with $\|AX - B\|_2 \leq \beta$. If we can now find a rank k matrix Z with orthonormal columns such that

$$\|ZZ^{\mathrm{T}}\frac{AA^+BM}{\beta} - \frac{AA^+BM}{\beta}\|_2 \leq (1+\varepsilon)\sigma_{k+1}\left(\frac{AA^+BM}{\beta}\right), \tag{B.6}$$

then

$$\begin{aligned}&\|ZZ^{\mathrm{T}}AA^+B(\beta^2 I - \Delta)^{-1/2} - AA^+B(\beta^2 I - \Delta)^{-1/2}\|_2 \\ &\leq \|ZZ^{\mathrm{T}}\frac{AA^+BM}{\beta} - \frac{AA^+BM}{\beta}\|_2 + \|(I - ZZ^{\mathrm{T}})\left(\frac{AA^+BM}{\beta} - AA^+B(\beta^2 I - \Delta)^{-1/2}\right)\|_2 \\ &\leq (1+\varepsilon)\sigma_{k+1}\left(\frac{AA^+BM}{\beta}\right) + \frac{\|AA^+B\|_2}{\beta}\delta \\ &\leq (1+\varepsilon)(1 + 2\|AA^+B\|_2(\delta/\beta)).\end{aligned}$$

The last inequality follows from (B.5). If δ is chosen to be less than $\varepsilon/4\kappa$ where $\kappa = \sigma_1(B)/\sigma_{k+1}(B)$,

Algorithm B.1: Oracle$_{\mathcal{M}'}$

```
Input: $v \in \mathbb{R}^d$, $\varepsilon_r > 0$
Output: $y \in \mathbb{R}^n$
/* Let $r(x)$ be the polynomial as in Lemma 6.6.2 */
1 $t \leftarrow \text{degree}(r)$
2 $\varepsilon_{\text{reg}} \leftarrow O\left(\varepsilon_r/\kappa\|r\|_1\right)$
3 $y \leftarrow 0$
4 $\text{Apx}_0 \leftarrow v$
5 for $i = 0, \ldots, t$ do
6     $y \leftarrow y + r_i \text{Apx}_i$
7     $\text{Apx}_{i+1} \leftarrow B^{\mathrm{T}} B \cdot \text{Apx}_i - B^{\mathrm{T}} \cdot (\textsc{HighPrecisionRegression}(A, B \cdot \text{Apx}_i, \varepsilon_{\text{reg}}))$
8 end
9 $y \leftarrow (\text{HighPrecisionRegression}(A, B \cdot y, \varepsilon_{\text{reg}}))/\beta$
```

then

$$
\begin{aligned}
&\|ZZ^{\mathrm{T}}AA^{+}B(\beta^2 I - \Delta)^{-1/2} - AA^{+}B(\beta^2 I - \Delta)^{-1/2}\|_2 \\
&\leq (1+\varepsilon)\left(1 + 2\|AA^{+}B\|_2(\delta/\beta)\right) \\
&\leq (1+\varepsilon)\left(1 + 2\frac{\|AA^{+}B\|_2}{\beta}\frac{\varepsilon\sigma_{k+1}(B)}{4\sigma_1(B)}\right) \\
&\leq 1 + 2\varepsilon
\end{aligned}
$$

as $\|AA^{+}B\|_2 \leq \sigma_1(B)$ and $\beta \geq (1+\varepsilon)\sigma_{k+1}(B)$. This implies that if $(1-x)^{-1/2}$ is approximated by a polynomial $r(x)$ uniformly in the interval $[0, 1/(1+\varepsilon)]$ with an error at most $\varepsilon/4\kappa$, and if matrix Z is an orthonormal basis for a space that spans a $1+\varepsilon$ rank k approximation in spectral norm for the matrix $AA^{+}B\frac{r(\Delta/\beta^2)}{\beta}$, then

$$
\|AA^{+}Z(AA^{+}Z)^{+}B - B\|_2 \leq (1+6\varepsilon)\beta = (1+O(\varepsilon))\text{OPT}.
$$

We obtain the proof by appropriately scaling ε. □

B.4.3 Proof of Lemma 6.6.3

Throughout the analysis, we assume $\|AA^{+}B\|_2 \geq \varepsilon\|B\|_2$. Suppose that $\|AA^{+}B\|_2 \leq \varepsilon\|B\|_2$. Let z be the top singular vector of matrix B. Then

$$
\begin{aligned}
\|B\|_2^2 &= \|Bz\|_2^2 \\
&= \|AA^{+}Bz\|_2^2 + \|(I - AA^{+})Bz\|_2^2 \\
&\leq \varepsilon^2\|B\|_2^2 + \|(I - AA^{+})Bz\|_2^2.
\end{aligned}
$$

Algorithm B.2: Oracle$_{\mathscr{M}'\mathscr{T}}$

```
    Input: $v \in \mathbb{R}^d$, $\varepsilon_{\mathrm{r}} > 0$
    Output: $y \in \mathbb{R}^n$
    /* Let $r(x)$ be the polynomial as in Lemma 6.6.2                    */
1   $t \leftarrow \mathrm{degree}(r)$
2   $\varepsilon_{\mathrm{reg}} \leftarrow O\left(\varepsilon_{\mathrm{r}}/\kappa\|r\|_1\right)$
3   $y \leftarrow 0$
4   $\mathrm{Apx}_0 \leftarrow B^{\mathrm{T}} \cdot (\textsc{HighPrecisionRegression}(A, v, \varepsilon_{\mathrm{reg}}))$
5   for $i = 0, \ldots, t$ do
6   |   $y \leftarrow y + r_i \mathrm{Apx}_i$
7   |   $\mathrm{Apx}_{i+1} \leftarrow B^{\mathrm{T}}B \cdot \mathrm{Apx}_i - B^{\mathrm{T}} \cdot (\textsc{HighPrecisionRegression}(A, B \cdot \mathrm{Apx}_i, \varepsilon_{\mathrm{reg}}))$
8   end
9   $y \leftarrow y/\beta$
```

Thus, $\|(I - AA^+)B\|_2^2 \geq \|(I - AA^+)Bz\|_2^2 \geq (1 - \varepsilon^2)\|B\|_2^2$. Therefore, $\mathrm{OPT} \geq \sqrt{1 - \varepsilon^2}\|B\|_2$ which implies $\|B\|_2 \leq (1/\sqrt{1 - \varepsilon^2})\mathrm{OPT} \leq (1 + \varepsilon)\mathrm{OPT}$ for $\varepsilon \leq 1/2$. Thus, $\|A(0) - B\|_2 \leq (1 + \varepsilon)\mathrm{OPT}$ and hence we have a trivial $(1 + \varepsilon)$-approximate solution. Thus, we can assume $\|AA^+B\|_2 \geq \varepsilon\|B\|_2$.

Based on Theorem 6.2.1, we compute approximate projections onto the column span of A. The following lemma states that a matrix-vector product with the matrix (Δ/β^2) can be approximated well.

Lemma B.4.5. *Given an arbitrary vector $v \in \mathbb{R}^d$, we can compute a vector $y \in \mathbb{R}^d$ such that*

$$\|y - (1/\beta^2)\Delta v\|_2 \leq \varepsilon_{\mathrm{reg}}\kappa\|v\|_2$$

in time $O(\mathrm{nnz}(B) + (\mathrm{nnz}(A) + c^2)\log(1/\varepsilon_{\mathrm{reg}}))$.

Proof. Recall that $\Delta = B^{\mathrm{T}}(I - AA^+)B$. Therefore, for a vector v, $\Delta v = B^{\mathrm{T}}Bv - B^{\mathrm{T}}AA^+Bv$. After computing Bv exactly, we can compute $\widetilde{y}$ by Theorem 6.2.1 in $O((\mathrm{nnz}(A) + c^2)\log(1/\varepsilon_{\mathrm{reg}}))$ time such that

$$\|AA^+Bv - \widetilde{y}\|_2 \leq \varepsilon_{\mathrm{reg}}\|(I - AA^+)Bv\|_2.$$

Let $y = B^{\mathrm{T}}Bv - B^{\mathrm{T}}\widetilde{y}$, which can be computed in $O(\mathrm{nnz}(B))$ time. Then $\Delta v - y = B^{\mathrm{T}}(\widetilde{y} - AA^+Bv)$, which implies $\|\Delta v - y\|_2 \leq \|B\|_2\|\widetilde{y} - AA^+Bv\|_2 \leq \varepsilon_{\mathrm{reg}}\|B\|_2\|(I - AA^+)B\|_2\|v\|_2$.

Thus, given a vector v, we can compute $(\Delta/\beta^2)v$ up to an error of

$$\varepsilon_{\mathrm{reg}}\|B\|_2\|(I - AA^+)B\|_2\|v\|_2/\beta^2 \leq \varepsilon_{\mathrm{reg}}\kappa\|v\|_2,$$

since $\beta \geq \max(\|(I - AA^+)B\|_2, \sigma_{k+1}(B))$. □

Lemma B.4.6. *Given an arbitrary vector $v \in \mathbb{R}^d$, for matrix $M = r\left(\Delta/\beta^2\right) = \sum_{j=0}^{t} r_j \left(\Delta/\beta^2\right)^j$ where the degree $t = O((1/\sqrt{\varepsilon})\log(\kappa/\varepsilon))$ and $\|r\|_1 = O((1 + \sqrt{2})^{O(\sqrt{1/\varepsilon}\log(\kappa/\varepsilon))}\log(\kappa/\varepsilon))$, we can compute a* *vector y such that $\|Mv - y\|_2 \leq \varepsilon_\mathrm{r}\|v\|_2$ in time*

$$O\left(t \cdot \left(\mathrm{nnz}(B) + (\mathrm{nnz}(A) + c^2)\log\left(\kappa\|r\|_1/\varepsilon_\mathrm{r}\right)\right)\right).$$

Proof. Let $\mathrm{Apx}_0 \coloneqq v$ and for $i \geq 1$, define Apx_i to be the approximation computed for the product $(\Delta/\beta^2)\mathrm{Apx}_{i-1}$ by Lemma B.4.5. Define

$$E_i \coloneqq \|(\Delta/\beta^2)^i v - \mathrm{Apx}_i\|_2.$$

We have the following recurrence

$$\begin{aligned} E_i = \left\|\left(\frac{\Delta}{\beta^2}\right)^i v - \mathrm{Apx}_i\right\|_2 &\leq \|(\Delta/\beta^2)^i v - (\Delta/\beta^2)Apx_{i-1}\|_2 + \|(\Delta/\beta^2)\mathrm{Apx}_{i-1} - \mathrm{Apx}_i\|_2 \\ &\leq \|(\Delta/\beta^2)\|_2 E_{i-1} + \varepsilon_\mathrm{reg}\kappa\|\mathrm{Apx}_{i-1}\|_2 \\ &\leq \|(\Delta/\beta^2)\|_2 E_{i-1} + \varepsilon_\mathrm{reg}\kappa \cdot \left(\|\Delta/\beta^2\|_2^{i-1}\|v\|_2 + E_{i-1}\right) \\ &\leq (\|\Delta\|_2/\beta^2 + \varepsilon_\mathrm{reg}\kappa)E_{i-1} + \varepsilon_\mathrm{reg}\kappa\|\Delta/\beta^2\|_2^{i-1}\|v\|_2. \end{aligned}$$

As $\beta \geq (1+\varepsilon)\|(I - AA^+)B\|_2$, we have that $\|\Delta/\beta^2\|_2 \leq 1/(1+\varepsilon)^2$. If $\varepsilon_\mathrm{reg}\kappa \leq \varepsilon/4$, then $\|\Delta/\beta^2\|_2 + \varepsilon_\mathrm{reg}\kappa \leq 1/(1+\varepsilon)^2 + \varepsilon/4 \leq 1/(1+\varepsilon)$. Therefore,

$$E_i \leq \frac{E_{i-1}}{1+\varepsilon} + \frac{\varepsilon_\mathrm{reg}\kappa}{(1+\varepsilon)^{2(i-1)}}\|v\|_2.$$

This implies upon solving the recurrence that

$$E_i \leq \varepsilon_\mathrm{reg}\kappa\|v\|_2$$

for all i. Then

$$\begin{aligned} \|Mv - \sum_{j=0}^{t} r_j \mathrm{Apx}_j\|_2 &\leq \sum_{j=0}^{t} |r_j| \|(\Delta/\beta^2)^j v - \mathrm{Apx}_j\|_2 \\ &\leq \sum_{j=0}^{t} |r_j| E_j \leq \varepsilon_\mathrm{reg}\kappa\|v\|_2 \sum_{j=0}^{t} |r_j| = \varepsilon_\mathrm{reg}\kappa\|v\|_2\|r\|_1. \end{aligned}$$

So for any arbitrary vector v, we can compute a vector y such that

$$\|Mv - y\|_2 \leq \varepsilon_\mathrm{r}\|v\|_2$$

by setting $\varepsilon_\mathrm{reg} = O(\frac{\varepsilon_\mathrm{r}}{\kappa\|r\|_1}) \leq \varepsilon/4\kappa$ for all t approximate products and thus y can be computed by

Lemma B.4.5 in time

$$O(t \cdot (\mathrm{nnz}(B) + (\mathrm{nnz}(A) + r^2) \log(\kappa \|r\|_1/\varepsilon_r))).$$

This conclues the proof of the lemma. □

Thus, for an arbitrary vector v, we can compute a vector y such that $\|Mv - y\|_2 \leq \varepsilon_r \|v\|_2$.

Proof of Lemma 6.6.3. Recall that $\mathcal{M}' = (AA^+BM)/\beta$, $\|M\|_2 \leq 2/\sqrt{\varepsilon}$ and $\sigma_{\min}(M) \geq 1/2$ from Lemma 6.6.2. We have $\|AA^+BM\|_2 \geq \|AA^+B\|_2 \sigma_{\min}(M) \geq \|AA^+B\|_2/2 \geq \varepsilon\|B\|_2/2$ where the last inequality follows from our assumption that $\|AA^+B\|_2 \geq \varepsilon\|B\|_2$. Thus, $\|\mathcal{M}'\|_2 \geq \varepsilon\|B\|_2/2\beta \geq \varepsilon/4$ as $\beta \leq (1+\varepsilon)\|B\|_2$.

Now we show how to compute approximations to $\mathcal{M}'v$ and $\mathcal{M}'Tv'$ for arbitrary vectors v, v'.

To compute an approximation to $\mathcal{M}'v$, we first obtain a vector y_1 using the above lemma such that $\|Mv - y_1\|_2 \leq \varepsilon_r\|v\|_2$. Then we compute the product By_1 exactly in time $O(\mathrm{nnz}(B))$. Thereafter, we compute a vector y_2 by Theorem 6.2.1 such that

$$\|y_2 - AA^+By_1\|_2 \leq \varepsilon_{\mathrm{reg}}\|(I - AA^+)By_1\|_2 \leq \varepsilon_{\mathrm{reg}}\|(I - AA^+)B\|_2\|y_1\|_2.$$

We also have $\|AA^+By_1 - AA^+BMv\|_2 \leq \varepsilon_r\|AA^+B\|_2\|v\|_2$. Therefore, by the triangle inequality, $\|AA^+BMv - y_2\|_2 \leq \varepsilon_r\|AA^+B\|_2\|v\|_2 + \varepsilon_{\mathrm{reg}}\|(I - AA^+)B\|_2\|y_1\|_2$. Hence,

$$\begin{aligned}\|\mathcal{M}'v - (y_2/\beta)\|_2 &\leq \varepsilon_r\frac{\|AA^+B\|_2}{\beta}\|v\|_2 + \varepsilon_{\mathrm{reg}}\|y_1\|_2 \leq \varepsilon_r\kappa\|v\|_2 + \varepsilon_{\mathrm{reg}}(\varepsilon_r\|v\|_2 + \|M\|_2 v)\\ &\leq \varepsilon_r(\kappa+1)\|v\|_2 + \frac{2\varepsilon_{\mathrm{reg}}}{\sqrt{\varepsilon}}\|v\|_2.\end{aligned}$$

Thus, if $\varepsilon_r = O(\varepsilon_f\varepsilon/\kappa)$ and $\varepsilon_{\mathrm{reg}} = O(\varepsilon_f\varepsilon^{3/2})$, we have that $\|\mathcal{M}'v - (y_2/\beta)\|_2 \leq \varepsilon_f\varepsilon\|v\|_2 \leq \varepsilon_f\|\mathcal{M}'\|_2\|v\|_2$. Therefore, a vector y_2/β can be computed in time $O(t \cdot (\mathrm{nnz}(B) + (\mathrm{nnz}(A) + c^2)\log\left(\frac{\kappa^2\|r\|_1}{\varepsilon_f\varepsilon}\right)) + O((\mathrm{nnz}(A) + c^2)\log(\frac{1}{\varepsilon_f\varepsilon}))$.

Now we compute an approximation to $\mathcal{M}'Tv = (M^{\mathrm{T}}B^{\mathrm{T}}AA^+/\beta)v$ for an arbitrary vector v. We first compute a vector y_1 such that

$$\|AA^+v - y_1\|_2 \leq \varepsilon_{\mathrm{reg}}\|(I - AA^+)v\|_2 \leq \varepsilon_{\mathrm{reg}}\|v\|_2.$$

Then we compute $B^{\mathrm{T}}y_1$ exactly. Then we compute a vector y_2 such that $\|MB^{\mathrm{T}}y_1 - y_2\|_2 \leq \varepsilon_r\|B^{\mathrm{T}}y_1\|_2 \leq \varepsilon_r\|B\|_2(1+\varepsilon_{\mathrm{reg}})\|v\|_2$. We further have

$$\|MB^{\mathrm{T}}AA^+v - MB^{\mathrm{T}}y_1\|_2 \leq \varepsilon_{\mathrm{reg}}\|MB^{\mathrm{T}}\|_2\|v\|_2 \leq \varepsilon_{\mathrm{reg}}\frac{2\|B\|_2}{\sqrt{\varepsilon}}\|v\|_2.$$

Thus,

$$\|y_2 - MB^{\mathrm{T}}AA^{+}v\|_2 \leq 2\varepsilon_{\mathrm{r}}\|B\|_2\|v\|_2 + \varepsilon_{\mathrm{reg}}\frac{2\|B\|_2}{\sqrt{\varepsilon}}\|v\|_2$$

and hence

$$\|y_2/\beta - \mathcal{M}'Tv\|_2 \leq 2\varepsilon_{\mathrm{r}}\kappa\|v\|_2 + \varepsilon_{\mathrm{reg}}\frac{2\kappa}{\sqrt{\varepsilon}}\|v\|_2$$

Now picking $\varepsilon_{\mathrm{r}} = O(\varepsilon_{\mathrm{f}}\varepsilon/\kappa)$ and $\varepsilon_{\mathrm{reg}} = O(\varepsilon_{\mathrm{f}}\varepsilon^{3/2}/\kappa)$, we obtain that

$$\|(y_2/\beta) - \mathcal{M}'Tv\|_2 \leq \varepsilon_{\mathrm{f}}\varepsilon\|v\|_2 \leq \varepsilon_{\mathrm{f}}\|\mathcal{M}'\|_2\|v\|_2.$$

Thus, this approximation can be computed in time $O(t \cdot (\mathrm{nnz}(B) + (\mathrm{nnz}(A) + c^2)\log\left(\frac{\kappa^2\|r\|_1}{\varepsilon_{\mathrm{f}}\varepsilon}\right))) + O((\mathrm{nnz}(A) + c^2)\log(\frac{\kappa}{\varepsilon_{\mathrm{f}}\varepsilon}))$. It follows that given an accuracy parameter ε_{f}, we can compute approximate matrix-vector products with $\mathcal{M}'$ and $\mathcal{M}'T$ in time at most

$$\begin{aligned}T(\varepsilon_{\mathrm{f}}) = O(t \cdot (\mathrm{nnz}(B) + (\mathrm{nnz}(A) + c^2)\log\left(\kappa(B)^2\|r\|_1/(\varepsilon_{\mathrm{f}}\varepsilon)\right)))\\ + O((\mathrm{nnz}(A) + c^2)\log(\kappa(B)/(\varepsilon_{\mathrm{f}}\varepsilon))).\end{aligned}$$

□

B.4.4 Proof of Theorem 6.6.4

Proof. From Lemma 6.6.2,

$$\sigma_1(\mathcal{M}') \leq \sigma_1\left(\frac{AA^{+}B}{\beta}\right)\|M\|_2 \leq \sigma_1\left(\frac{AA^{+}B}{\beta}\right)\frac{2}{\sqrt{\varepsilon}}$$

and

$$\sigma_{k+1}(\mathcal{M}') \geq \sigma_{k+1}(AA^{+}B/\beta) \cdot \sigma_{\min}(M) \geq \sigma_{k+1}(AA^{+}B/\beta)(1/2).$$

Therefore, $\kappa(\mathcal{M}') \leq \sigma_1(AA^{+}B/\beta)(2/\sqrt{\varepsilon})/\sigma_{k+1}(AA^{+}B/\beta)/2 \leq (4/\sqrt{\varepsilon})\kappa(AA^{+}B)$. By Theorem 6.5.1, we can compute a matrix $Z \in \mathbb{R}^{n\times k}$ such that $\|(I - ZZ^{\mathrm{T}})\mathcal{M}'\|_2 \leq (1 + 2\varepsilon)\sigma_{k+1}(\mathcal{M}')$ in time

$$T\left(\frac{\varepsilon}{\kappa(\mathcal{M}')^{5q}k^{11}C^q}\right)qk + T\left(\frac{\varepsilon^2}{48\kappa(\mathcal{M}'^2(\sqrt{qk})k)}\right)qk,$$

where $q = O((1/\sqrt{\varepsilon})\log(d/\varepsilon))$. Thus, the total time required is

$$O\left(tqk \cdot \left(\mathrm{nnz}(B) + (\mathrm{nnz}(A) + c^2)\log\left(\frac{\kappa^2\|r\|_1\kappa(\mathcal{M}')^{5q}k^{11}C^q}{\varepsilon^2}\right)\right)\right).$$

As $\|r\|_1 = (1+\sqrt{2})^{O(1/\sqrt{\varepsilon}\log(\kappa/\varepsilon))}\log(\kappa/\varepsilon)/\varepsilon$ and $\kappa(\mathcal{M}') = \kappa(AA^+B)/\sqrt{\varepsilon}$, we obtain that the total time required is $O(tqk \cdot \mathrm{nnz}(B) + tqk \cdot (\frac{1}{\sqrt{\varepsilon}}\log(\kappa/\varepsilon)+q)\log(\frac{\kappa\cdot\kappa(\mathcal{M}')\cdot k}{\varepsilon}) \cdot (\mathrm{nnz}(A)+c^2))$. Substituting $t = O(\sqrt{1/\varepsilon}\log(\kappa/\varepsilon))$, we obtain that the total running time is

$$O\left(\left(\frac{\mathrm{nnz}(B)\cdot k}{\varepsilon} + \frac{\mathrm{nnz}(A)\cdot k}{\varepsilon^{1.5}} + \frac{c^2k}{\varepsilon^{1.5}}\right)\cdot \mathrm{polylog}(\kappa, \kappa(AA^+B), d, k, 1/\varepsilon)\right), \tag{B.7}$$

and there is an additional c^ω time for computing a preconditioner. By Lemmas 6.6.1 and 6.6.2, we obtain that

$$\|(AA^+Z)(AA^+Z)^+B - B\|_2 = \|ZZ^{\mathrm{T}}B - B\|_2 \le (1+O(\varepsilon))\mathrm{OPT}.$$

The equality is from the fact that Z is spanned by the columns of matrix A by Theorem 6.5.1, and therefore $AA^+Z = Z$. Thus, there exists a matrix $X_1 \in \mathbb{R}^{c\times k}$ such that $AX_1 = Z$ and the matrix X_1 can be computed in time $O((\mathrm{nnz}(A)+c^2)k + c^\omega)$ using sketching-based preconditioning techniques. Let $Y_1 = Z^{\mathrm{T}}B$, which can be computed in time $O(\mathrm{nnz}(B)\cdot k)$. Therefore,

$$\|AX_1Y_1 - B\|_2 = \|ZZ^{\mathrm{T}}B - B\|_2 \le (1+O(\varepsilon))\mathrm{OPT}.$$

Thus, $X_1 \cdot Y_1$ is a $(1+O(\varepsilon))$-approximation to the regression problem. By appropriately scaling ε, we obtain the proof. □

B.4.5 Proof of Lemma 6.6.5

Proof. Let $G \sim N(0,1)^{n\times(k+1)}$ and $F^{\mathrm{T}} \in \mathbb{R}^{(k+1)\times d}$ be a matrix with $k+1$ orthonormal rows. Let α be a parameter to be chosen later and $\widetilde{B} \coloneqq B + \alpha GF^{\mathrm{T}}$. For all matrices X, by the triangle inequality,

$$\|AX - \widetilde{B}\|_2 \in \|AX - B\|_2 \pm \alpha\|GF^{\mathrm{T}}\|_2.$$

With probability $\ge 9/10$, $\|G\|_2 \le 2\sqrt{n}$. Thus, $\|AX - \widetilde{B}\|_2 \in \|AX - B\|_2 \pm 2\alpha\sqrt{n}$. Therefore, if $\widetilde{X}$ is a $(1+\varepsilon)$-approximation to $\min_{\text{rank-}k\ X}\|AX - \widetilde{B}\|_2$, then $\|A\widetilde{X} - B\|_2 \le (1+\varepsilon)\mathrm{OPT} + 6\alpha\sqrt{n}$.

We now have $\sigma_1(AA^+\widetilde{B}) \le \|\widetilde{B}\|_2 \le \|B\|_2 + 2\alpha\sqrt{n}$ from the above discussion. We now lower bound $\sigma_{k+1}(AA^+\widetilde{B})$. Let U be an orthonormal basis for the columns of A. Therefore, $AA^+ = UU^{\mathrm{T}}$.

$$\begin{aligned}
\sigma_{k+1}(AA^+\widetilde{B}) &= \sigma_{k+1}(UU^{\mathrm{T}}\widetilde{B}) \\
&= \sigma_{k+1}(U^{\mathrm{T}}\widetilde{B}) \\
&= \sigma_{k+1}(U^{\mathrm{T}}B + \alpha U^{\mathrm{T}}GF^{\mathrm{T}}) \\
&\ge \sigma_{k+1}(U^{\mathrm{T}}BFF^{\mathrm{T}} + \alpha U^{\mathrm{T}}GF^{\mathrm{T}}) \\
&\ge \sigma_{k+1}(U^{\mathrm{T}}BF + \alpha U^{\mathrm{T}}G).
\end{aligned}$$

As the rows of U^{T} are orthonormal, the matrix $G' = U^{\mathrm{T}}G$ is a matrix of i.i.d. normal random variables. Assuming A is of full rank, G' is a $c \times (k+1)$ matrix. Assuming $c \geq k+1$, let E be the top $(k+1) \times (k+1)$ submatrix of $U^{\mathrm{T}}BF + \alpha G'$. Then E can be seen as a fixed $(k+1) \times (k+1)$ matrix where each entry is perturbed by a Gaussian random variable of variance α^2. From Theorem 2.2 of [VT07], we obtain that $\sigma_{\min}(E) \geq \alpha/(C\sqrt{k})$ for a constant C with probability $\geq 9/10$. Thus, $\sigma_{k+1}(U^{\mathrm{T}}BF + \alpha U^{\mathrm{T}}G) \geq \sigma_{\min}(E) \geq \alpha/(C\sqrt{k})$.

Thus, $\sigma_1(AA^+\widetilde{B})/\sigma_{k+1}(AA^+\widetilde{B}) \leq (\|B\|_2 + 2\alpha\sqrt{n})/(\alpha/(C\sqrt{k}))$. For $\alpha = \frac{\varepsilon\sigma_{k+1}(B)}{(6\sqrt{n})}$, we obtain that

$$\sigma_1(AA^+\widetilde{B})/\sigma_{k+1}(AA^+\widetilde{B}) \leq \frac{Cn}{\varepsilon}\kappa$$

for a constant C with probability $\geq 4/5$. Also, if $\widetilde{X}$ is a $(1+\varepsilon)$-approximation as mentioned above, $\|A\widetilde{X} - B\|_2 \leq (1+\varepsilon)\mathrm{OPT} + \varepsilon\sigma_{k+1}(B) \leq (1+2\varepsilon)\mathrm{OPT}$. We obtain the proof by scaling ε appropriately. □

Appendix C

Deferred Details from Chapter 10

C.1 Nisan's Pseudorandom Generator

We say that a *randomized* program uses space w with a block size n if it accepts its random bits as an n bit block at a time and uses at most space w between the different blocks of random bits. Such programs can be modeled as a finite state machine over at most 2^w states, taking an input string over the alphabet $\{0,1\}^n$. Nisan [Nis92] constructed a pseudorandom generator which requires only a small uniform random seed that "fools" a space w program with a block size n.

Let $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k$ be independent hash functions drawn from a 2-wise independent hash family $\mathscr{H} = \{h : \{0,1\}^n \rightarrow \{0,1\}^n\}$. These hash functions together with $\boldsymbol{x} \in \{0,1\}^n$, sampled uniformly at random, serve as the *seed* of the generator $G_k : \{0,1\}^n \rightarrow \{0,1\}^{2^k \cdot n}$, defined recursively as follows:

$$G_0(x) \coloneqq x$$
$$G_k(x, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k) \coloneqq G_{k-1}(x, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{k-1}) \circ G_{k-1}(\boldsymbol{h}_k(x), \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{k-1}),$$

where $\circ$ denotes the string concatenation. For a given choice of $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k$, define the distribution $G_k(*, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)$ over bitstrings of length $2^k \cdot n$ to be the distribution of $G_k(\boldsymbol{x}, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)$ for random $\boldsymbol{x} \in \{0,1\}^n$. Nisan showed that for any fixed FSM with at most 2^w states over alphabet $\{0,1\}^n$, with high probability over the hash functions $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k$, the distribution $G_k(*, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)$ is indistinguishable from the uniform distribution over $\{0,1\}^{2^k \cdot n}$. The power of Nisan's generator is summarized by the following lemma (using notation from section 10.3):

Lemma C.1.1. *There exists a constant $c > 0$ such that given integers n and $w \leq cn$ and parameter $k \leq cn$, for any FSM Q with 2^w states, if $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k : \{0,1\}^n \rightarrow \{0,1\}^n$ are drawn independently from a 2-wise independent hash family, then with probability $\geq 1 - 2^{-cn}$,*

$$\|Q(G_k(*, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)) - Q((U_n)^{2^k})\| \leq 2^{-cn}$$

where $\|M\| \coloneqq \max_i \sum_j |M_{ij}|$.

Note that $\|M\| = \max_{x \neq 0} \|Mx\|_\infty / \|x\|_\infty$ where $\|x\|_\infty = \max_i |x_i|$. We therefore have that for any two matrices A and B, $\|A+B\| \le \|A\|+\|B\|$ and $\|AB\| \le \|A\|\|B\|$. Therefore, we obtain that with probability $\ge 1 - 2^{-cn}$ over the hash functions $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k$, we have that the total variation distance between the distribution of final state using a random string drawn from $(U_n)^{2^k}$ and a random string drawn from $G_k(*, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_k)$ is at most 2^{-cn}.

Specifically, for a $w = O(\log d)$ space algorithms using $\text{poly}(d)$ random bits, we have that we can use Nisan's Generator with $n, k = O(\log d)$. The time to evaluate a block of n random bits is then $\Omega(k) = \Omega(\log d)$ in the Word RAM model as the k hash functions have to be applied sequentially to the random seed. Using our new pseudorandom generator, which we call HashPRG, we show that we can set $k = O(1)$ at the expense of using more space to store the hash functions.

C.2 Finding Heavy Entries

See Section 10.6 for the definition of CountSketch data structure. Note that $[t]$ denotes the range of locations the coordinate gets hashed into and r denotes the number of repetitions. Further, for each $\ell \in [d]$, $\hat{x}_\ell$ defined in (10.10) denotes our estimate for the value of coordinate x_ℓ.

Jowhari, Sağlam and Tardos [JST11] show that if $r = O(\log d)$, then with probability $\ge 1 - 1/\text{poly}(d)$, for all ℓ,

$$|x_\ell - \hat{x}_\ell| \le \frac{\|x\|_p}{t^{1/p}}.$$

By picking $t = (\phi/10)^{-p}$ we obtain that with probability $1 - 1/\text{poly}(d)$, for all $\ell \in [d]$, $|x_\ell - \hat{x}_\ell| \le (\phi/10)\|x\|_p$. The algorithm uses $O((\phi/10)^{-p} \log^2 d)$ bits of space and has an update time of $O(\log d)$ per stream element. Condition on the event that for all $\ell \in [d]$, $|x_\ell - \hat{x}_\ell| \le (\phi/10)\|x\|_p$ for all $\ell \in [d]$.

Concurrently, run the algorithm of [KNW10] with $\varepsilon = 1/4$ to obtain a value v such that with probability $\ge 99/100$

$$(9/10)\|x\|_p \le v \le (11/10)\|x\|_p.$$

Note that for constant ε, their algorithm uses $O(\log d)$ bits of space and has an update time of $O(1)$ per stream element in the Word RAM model. Condition on this event as well.

Now, let L' be the set returned by heavy-hitters algorithm of [LNNT16] with parameter ϕ. Their algorithm uses $O(\phi^{-p} \log^2(d))$ bits of space and has an update time of $O(\log d)$ per stream element. At the end of processing the stream, in time $O(\phi^{-p}\, \text{poly}(\log d))$, they return a set L' satisfying $|L'| =$

$O(\phi^{-p})$ and

$$L' \supseteq \{\ell \mid |x_\ell| \ge \phi\|x\|_p\}.$$

The set L' contains all the heavy-hitters and may contain additional coordinates as well. To filter the list L', we use the estimates $\hat{x}_\ell$ given by the CountSketch data structure. Define

$$L = \{i \in L' \mid |\hat{x}_i| \ge (8/10)\phi v\}.$$

Conditioned on the correctness of L', $\hat{x}_\ell$ and the estimate v, we prove properties about the set L. If $|x_\ell| \ge \phi\|x\|_p$, then $|\hat{x}_\ell| \ge (9\phi/10)\|x\|_p \ge (9\phi/11)v \ge (8\phi/10)v$. Therefore, $\ell \in L$. On the other hand, if $\ell \in L$ then

$$|x_\ell| \ge |\hat{x}_\ell| - (\phi/10)\|x\|_p \ge (8/10)\phi v - (\phi/10)\|x\|_p \ge (6\phi/10)\|x\|_p.$$

As $|x_\ell - \hat{x}_\ell| \le (\phi/10)\|x\|_p$ for all ℓ and $|x_\ell| \ge (6\phi/10)\|x\|_p$ for all $\ell \in L$, we also obtain that for all $\ell \in L$, $\text{sign}(x_\ell) = \text{sign}(\hat{x}_\ell)$. As the list L' has size at most $O(\phi^{-p})$, the post-processing can be performed in time $O(\phi^{-p}\,\text{poly}(\log d))$. Thus, we over all have the following lemma.

Lemma C.2.1. *Given a stream of updates $(i_1, v_1), \ldots, (i_m, v_m) \in [d] \times \{-M, \ldots, M\}$ for $m, M \le \text{poly}(d)$, a parameter ϕ and $p \in (0, 2)$, there is a streaming algorithm that uses $O(\phi^{-p}\log^2(d))$ bits of space and has an update time of $O(\log d)$ per stream element and outputs a set $L \subseteq [d]$ at the end of the stream that with probability $\ge 9/10$ satisfies:*

1. *$L \supseteq \{\ell \in [d] \mid |x_\ell| \ge \phi\|x\|_p\}$.*
2. *For all $\ell \in L$, $|x_\ell| \ge (6\phi/10)\|x\|_p$.*
3. *For all $\ell \in L$, the algorithm also outputs $\text{sign}(x_\ell)$.*

At the end of the stream, the algorithm takes only $O(\phi^{-p}\,\text{poly}(\log d))$ time to compute the set L.

Appendix D

Deferred Proofs from Chapter 11

D.1 Omitted Proofs from Section 11.3

D.1.1 Proof of Lemma 11.3.4

Let i^* be the largest index such that $\mathrm{rank}(B_{1:i}) = k$. We note $\mathrm{rank}(B_{1:i^*+1}) = k + 1$. We now separate the sum of online rank-k ridge leverage scores as

$$\sum_{i=1}^{n} \tau_i^{\mathrm{OL},k}(B) = \sum_{i=1}^{i^*+1} \tau_i^{\mathrm{OL},k}(B) + \sum_{i=i^*+2}^{n} \tau_i^{\mathrm{OL},k}(B)$$

and bound both the terms separately. Let RI[1] $\subseteq [i^* + 1]$ be the set of coordinates i such that $\mathrm{rank}(B_{1:i}) > \mathrm{rank}(B_{1:i-1})$. Note that $|\mathrm{RI}| \le k + 1$. By definition of the rank-k ridge leverage scores, we have for all $i \in \mathrm{RI}$, $\tau_i^{\mathrm{OL},k}(B) = 1$. Now consider an $i < i^* + 1$ and $i \notin \mathrm{RI}$. We have

$$\tau_i^{\mathrm{OL},k}(B) = \min(1, b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}} B_{1:i-1})^+ b_i).$$

We define $\sigma_{\min,\mathrm{RI}} \coloneqq \min_{i \in \mathrm{RI}} \sigma_{\min}(B_{1:i})$ where $\sigma_{\min}(\cdot)$ is used to denote the smallest *nonzero* singular value of the matrix B. We note that for all $i \in \mathrm{RI}$, $\|b_i\|_2 \ge \sigma_{\min,\mathrm{RI}}$.

Now consider $i < i^* + 1$ and $i \notin \mathrm{RI}$. Note that $b_i \in \mathrm{rowspace}(B_{1:i-1})$.

Claim D.1.1. *For $\sigma_{\min,RI}$ defined as above, the following hold for all $i \notin RI$:*

1.

$$b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}(B_{1:i-1}))^+ b_i \le 2 \cdot b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}} B_{1:i-1} + \sigma_{\min,RI}^2 \cdot I)^+ b_i.$$

[1]for **R**ank **I**ncrease

2.

$$\tau_i^{\mathrm{OL},k}(B) = \min(1, b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}(B_{1:i-1}))^{+}b_i) \leq 2 \cdot \min(1, b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}B_{1:i-1} + \sigma_{\min,RI}^2 \cdot I)^{+}b_i).$$

Proof. Let $U\Sigma V^{\mathrm{T}}$ be the "thin" singular value decomposition of the matrix B_{i-1}. It is easy to see that $\sigma_{\min}(B_{i-1}) \geq \sigma_{\min,\mathrm{RI}}$. Since $i \notin \mathrm{RI}$, we have $b_i \in \mathrm{rowspace}(B_{1:i-1})$ which then implies that we can write $b_i = V \cdot z$ for some z and therefore

$$b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}(B_{1:i-1}))^{+}b_i = z^{\mathrm{T}}\Sigma^{-2}z^{\mathrm{T}}.$$

We can also write

$$((B_{1:i-1})^{\mathrm{T}}B_{1:i-1} + \sigma_{\min,\mathrm{RI}}^2 \cdot I)^{+} = V(\Sigma^2 + \sigma_{\min,\mathrm{RI}}^2 \cdot I)^{-1}V^{\mathrm{T}} + \frac{1}{\sigma_{\min,\mathrm{RI}}^2}(I - VV^{\mathrm{T}})$$

from which we obtain

$$b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}B_{1:i-1} + \sigma_{\min,\mathrm{RI}}^2 \cdot I)^{+}b_i = z^{\mathrm{T}}(\Sigma^2 + \sigma_{\min,\mathrm{RI}}^2 \cdot I)^{-1}z \geq \frac{1}{2} \cdot z^{\mathrm{T}}\Sigma^{-2}z = \frac{1}{2}b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}(B_{1:i-1}))^{+}b_i,$$

where the last inequality follows from the fact that $0 \prec \Sigma^2 + \sigma_{\min,\mathrm{RI}}^2 \cdot I \preceq 2 \cdot \Sigma^2$.

Note that the second claim directly follows from the first. □

For $i \in \mathrm{RI}$, we prove the following:

Claim D.1.2. *For all $i \in RI$,*

$$1 = \tau_i^{\mathrm{OL},k}(B) \leq b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}B_{1:i-1} + \sigma_{\min,RI}^2 \cdot I)^{+}b_i.$$

Proof. Let $b_i^{\perp}$ be the projection of b_i away from $\mathrm{rowspace}(B_{1:i-1})$. Note that $b_i^{\perp}$ is in the rowspace of $B_{1:i}$ and therefore

$$|\langle b_i, b_i^{\perp}\rangle| = \|(B_{1:i}) \cdot b_i^{\perp}\|_2 \geq \sigma_{\min,\mathrm{RI}} \cdot \|b_i^{\perp}\|_2$$

which implies

$$\frac{|\langle b_i, b_i^{\perp}\rangle|^2}{\|B_{1:i-1} \cdot b_i^{\perp}\|_2^2 + \sigma_{\min,\mathrm{RI}}^2\|b_i^{\perp}\|_2^2} \geq \frac{\sigma_{\min,\mathrm{RI}}^2\|b_i^{\perp}\|_2^2}{0 + \sigma_{\min,\mathrm{RI}}^2\|b_i^{\perp}\|_2} \geq 1.$$ □

Thus, for all $i < i^* + 1$, we have

$$\tau_i^{\mathrm{OL},k}(B) \leq 2 \cdot \min(1, b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}B_{1:i-1} + \sigma_{\min,\mathrm{RI}}^2 \cdot I)^{+}b_i).$$

Hence, it suffices to bound $\sum_{i=1}^{i^*+1} \min(1, b_i^{\mathrm{T}}((B_{1:i-1})^{\mathrm{T}}B_{1:i-1} + \sigma_{\min,\mathrm{RI}}^2 \cdot I)^{+}b_i)$. By Theorem 2.2 of

[CMP16], we can bound this quantity by $O(k \log \|B_{1:i^*+1}\|_2/\sigma_{\min,\mathrm{RI}})$. Hence,

$$\sum_{i=1}^{i^*+1} \tau_i^{\mathrm{OL},k}(B) = O\left(k \log \frac{\|B_{1:i^*+1}\|_2}{\sigma_{\min,\mathrm{RI}}}\right).$$

We now want to bound

$$\sum_{i=i^*+2}^{n} \tau_i^{\mathrm{OL},k}(B) = \sum_{i=i^*+2}^{n} \min(1, b_i^{\mathrm{T}}(B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \frac{\|B_{1:i-1} - [B_{1:i-1}]_k\|_{\mathrm{F}}^2}{k} \cdot I)^{-1}b_i). \tag{D.1}$$

[BDM+20] shows a bound on the $\sum_{i=1}^{n} \min(1, b_i^{\mathrm{T}}(B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \lambda I)^{-1}b_i)$ where $\lambda = \|B - [B]_k\|_{\mathrm{F}}^2/k$. The only difference in the above term we want to bound is that, instead of using a fixed λ for all the terms as in [BDM+20], we require an upper bound when each term has a different multiple of the identity matrix.

We will now state some useful facts, that let us use the upper bounds from [BDM+20] to bound the term in (D.1). Suppose α is such that $\alpha/2 \le \|B_{1:i-1} - [B_{1:i-1}]_k\|_{\mathrm{F}}^2/k \le \alpha$. Then, we have from the standard properties of the Löwner ordering that,

$$\frac{1}{2}B_{1:i-1}B_{1:i-1}^{\mathrm{T}} + \frac{\alpha}{2} \cdot I \preceq B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \frac{\alpha}{2} \cdot I \preceq B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \frac{\|B_{1:i-1} - [B_{1:i-1}]_k\|_{\mathrm{F}}^2}{k} \preceq B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \alpha \cdot I.$$

Since all the above matrices are positive definite, assuming $\alpha > 0$, we obtain that

$$2\left(B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \alpha \cdot I\right)^{-1} \succeq (B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \frac{\|B_{1:i-1} - [B_{1:i-1}]_k\|_{\mathrm{F}}^2}{k} \cdot I)^{-1} \succeq (B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \alpha \cdot I)^{-1}$$

and therefore,

$$\min(1, b_i^{\mathrm{T}}(B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \frac{\|B_{1:i-1} - [B_{1:i-1}]_k\|_{\mathrm{F}}^2}{k} \cdot I)^{-1}b_i) \le 2 \cdot \min(1, b_i^{\mathrm{T}}(B_{1:i-1}^{\mathrm{T}}B_{1:i-1} + \alpha \cdot I)^{-1}b_i). \tag{D.2}$$

We note that $\|B_{1:i^*+1} - [B_{1:i^*+1}]_k\|_{\mathrm{F}}^2 = \sigma_{\min}(B_{1:i^*+1})^2 \ge \sigma_{\min,\mathrm{RI}}^2$ where we used the fact that the rank of $B_{1:i^*+1}$ is exactly $k+1$. For $j = 1, \ldots$, let i_j be the largest i such that

$$\frac{\|B_{1:i-1} - [B_{1:i-1}]_k\|_{\mathrm{F}}^2}{k} \le 2^j \cdot \frac{\sigma_{\min,\mathrm{RI}}^2}{k}$$

and consider the intervals of integers, $(k+1 = i_0, i_1], (i_1, i_2], (i_2, i_3], \ldots$. We note that there are at most

$$O\left(\log \frac{\|B - [B]_k\|_\mathrm{F}^2}{\sigma_{\mathrm{min,RI}}^2}\right) = O\left(\log \frac{\|B\|_2}{\sigma_{\mathrm{min,RI}}}\right)$$

such non-empty intervals. Now consider an arbitrary interval $(i_j, i_{j+1}]$, and we will bound

$$\sum_{i \in (i_j, i_{j+1}]} \min(1, b_i^\mathrm{T}(B_{1:i-1}^\mathrm{T} B_{1:i-1} + \frac{\|B_{1:i-1} - [B_{1:i-1}]_k\|_\mathrm{F}^2}{k} \cdot I)^{-1} b_i).$$

Setting $\alpha = 2^{j+1}\sigma_{\mathrm{min,RI}}^2/k$ in (D.2), we get

$$\begin{aligned}
&\sum_{i \in (i_j, i_{j+1}]} \min(1, b_i^\mathrm{T}(B_{1:i-1}^\mathrm{T} B_{1:i-1} + \frac{\|B_{1:i-1} - [B_{1:i-1}]_k\|_\mathrm{F}^2}{k} \cdot I)^{-1} b_i) \\
&\leq 2 \cdot \sum_{i \in (i_j, i_{j+1}]} \min(1, b_i^\mathrm{T}(B_{1:i-1}^\mathrm{T} B_{1:i-1} + 2^{j+1}\frac{\sigma_{\mathrm{min,RI}}^2}{k} \cdot I)^{-1} b_i)
\end{aligned}$$

and since by definition $\frac{\|B_{1:i_{j+1}-1} - [B_{1:i_{j+1}-1}]_k\|_\mathrm{F}^2}{k} \leq 2^{j+1}\frac{\sigma_{\mathrm{min,RI}}^2}{k}$, we further obtain

$$\begin{aligned}
&\sum_{i \in (i_j, i_{j+1}]} \min(1, b_i^\mathrm{T}(B_{1:i-1}^\mathrm{T} B_{1:i-1} + \frac{\|B_{1:i-1} - [B_{1:i-1}]_k\|_\mathrm{F}^2}{k} \cdot I)^{-1} b_i) \\
&\leq \sum_{i \in (i_j, i_{j+1}]} \min(1, m_i^\mathrm{T}(B_{1:i-1}^\mathrm{T} B_{1:i-1} + \frac{\|B_{1:i_{j+1}-1} - [B_{1:i_{j+1}-1}]_k\|_\mathrm{F}^2}{k} \cdot I)^{-1} m_i).
\end{aligned}$$

We can then finally use Lemma 2.11 of [BDM+20] to bound the above term by

$$k \log\left(1 + \frac{k\|B_{1:i_{j+1}-1}\|_2^2}{\|B_{1:i_{j+1}-1} - [B_{1:i_{j+1}-1}]_k\|_\mathrm{F}^2}\right) + k + 1 \leq k\log(1 + k\|B\|_2^2/\sigma_{\mathrm{min,RI}}^2) + k + 1$$

where we used the facts that $\|B_{1:i_{j+1}-1} - [B_{1:i_{j+1}-1}]_k\|_\mathrm{F}^2 \geq \|B_{i^*+1} - [B_{i^*+1}]_k\|_\mathrm{F}^2 \geq \sigma_{\mathrm{min,RI}}^2$ and $\|B_{1:i_{j+1}-1}\|_2^2 \leq \|B\|_2^2$. Overall, we get that

$$O(k\log(1 + k\|B\|_2/\sigma_{\mathrm{min,RI}})^2) = O(k\log(k \cdot \kappa)^2).$$

Appendix E

Deferred Details from Chapter 13

E.1 Gap in the analysis of [KVW14]

In Theorem 1.6 of [KVW14], the authors claim an F_k estimation algorithm that uses $\widetilde{O}(\varepsilon^{-3}(s^{k-1} + s^3)(\ln s)^3)$ bits of total communication. In the proof of Theorem 1.6, in the inequalities used to bound the quantity $\mathbf{E}(Y^2)/(\mathbf{E}(Y))^2$, the last inequality seems to use that $\rho_i \geq \beta/e^{\varepsilon}$, but the inequality holds only when $i \in S_\beta$ (in their notation). But the question of if $i \in S_\beta$ is exactly what they are trying to find out from the analysis and hence it cannot be assumed that $\rho_i \geq \beta/e^{\varepsilon}$.

E.2 Huber Loss Function

Given a parameter τ, the Huber loss function f is defined as $f(x) = x^2/(2\tau)$ if $|x| \leq \tau$ and $f(x) = |x| - \tau/2$ if $|x| \geq \tau$. In this work we consider only the values of $x \geq 0$. We will now examine various properties of the Huber loss function.

E.2.1 Super-Additivity

One can verify that the Huber loss function is convex and $f(0) = 0$. Consider arbitrary $x, y \geq 0$. Since f is convex in the interval $[0, x+y]$, we get

$$f(x) \leq \frac{x}{x+y} f(x+y) \quad \text{and} \quad f(y) \leq \frac{y}{x+y} f(x+y).$$

Adding both the inequalities, we get $f(x) + f(y) \leq f(x+y)$. Thus, we have the following lemma:

Lemma E.2.1. *If f is convex and $f(0) = 0$, then for any $x, y \geq 0$, we have $f(x+y) \geq f(x) + f(y)$.*

E.2.2 Bounding $c_f[s]$

We note that when f denotes the Huber loss function with parameter τ, the function $\sqrt{f}$ is concave on the interval $[0, \infty)$. Now consider arbitrary $x_1, \dots, x_s \geq 0$. By concavity of $\sqrt{f}$ in the interval $[0, x_1 + \cdots + x_s]$, we get

$$\sqrt{f(x_j)} \geq \frac{x_j}{x_1 + \cdots + x_s} \sqrt{f(x_1 + \cdots + x_s)}$$

for all $j = 1, \dots, s$. By adding all the inequalities,

$$\sqrt{f(x_1)} + \cdots + \sqrt{f(x_s)} \geq \sqrt{f(x_1 + \cdots + x_s)}$$

which implies

$$f(x_1 + \cdots + x_s) \leq \left(\sqrt{f(x_1)} + \cdots + \sqrt{f(x_s)}\right)^2$$

and therefore that $c_f[s] \leq s$ when f is the Huber loss function.

E.3 Derandomizing Exponential Random Variables using Nisan's PRG

Our algorithm for estimating higher-order correlations assumes that we have access to $O(n^k)$ independent exponential random variables, which raises the question how these are stored since the protocol later requires the values of these random variables. We now argue that Nisan's PRG [Nis92] can be used to derandomize the exponential random variables and that all the required exponential random variables can be generated using a short seed of length $O(k \log^2(n/\varepsilon))$.

Note that using $O(k \log(n/\varepsilon))$ bits of precision, we can sample from a discrete distribution that approximates the continuous exponential random variables up to a $1 \pm \varepsilon$ factor since by a simple union bound if we sample $O(n^k/\varepsilon^2)$ exponential random variables, then they all lie in the interval $[\text{poly}(\varepsilon) \cdot n^{-O(k)}, O(k \log n/\varepsilon)]$ with a $1 - 1/\text{poly}(n)$ probability. Let $b = O(k \log(n/\varepsilon))$. We can use b uniform bits to sample from this discrete distribution and store the sampled discrete random variables using b bits as well while ensuring that the discrete random variable has all the properties we use of the continuous random variable. Let $\boldsymbol{e}$ be the random variable drawn from this discrete distribution and let $\boldsymbol{e}_1, \dots, \boldsymbol{e}_n$ independent copies of this discrete random variable. Since the distribution of $\boldsymbol{e}$ is obtained by discretizing the continuous exponential random variable into powers of $1 + \varepsilon/4$, we have that $\max(f_i/\boldsymbol{e}_i)$ has the same distribution of $(\sum_i f_i)/\boldsymbol{e}$ up to a $1 \pm \varepsilon/4$ factor and with probability $\geq 1 - 1/\text{poly}(n)$, $\sum_i \boldsymbol{e}_i^{-1} f_i \leq (C \log^2 n) \cdot \max_i f_i/\boldsymbol{e}_i$ still holds with a slightly larger value of C.

To run the protocol for approximating higher-order correlations, we need to generate the same $r' = O(\binom{n}{k} \cdot k! \cdot \varepsilon^{-2})$ exponential random variables at all the servers and the coordinator. Suppose that the exponential random variables are generated using a random string of length $b \cdot r'$ as follows: we use the first $b \cdot \binom{n}{k} \cdot k!$ bits to generate the first set of exponential random variables, one for each coordinate of the form $(i_1, \ldots, i_k)$ for distinct $i_1, \ldots, i_k \in [n]$. We use the second set of $b \cdot \binom{n}{k} k!$ random bits to generate second set of exponential random variables and so on we generate $m = O(1/\varepsilon^2)$ sets of exponential random variables necessary for implementing the protocol. Let $\boldsymbol{e}^{(t)}_{(i_1,i_2,\ldots,i_k)}$ be the discrete exponential random variable corresponding to the coordinate $(i_1, \ldots, i_k)$ in the t-th set of random variables.

Let f be a vector with coordinates of the form $(i_1, \ldots, i_k)$ for distinct $i_1, \ldots, i_k \in [n]$. Now consider the following simple "small-space" algorithm Alg. It makes a pass on the length $b \cdot r$ string reading b blocks at a time and maintains the following counts:

1. `CountLess`: The number of values t such that

$$\max_{i_1,\ldots,i_k} (\boldsymbol{e}^{(t)}_{(i_1,\ldots,i_k)})^{-1} f_{(i_1,\ldots,i_k)} \geq (1+\varepsilon) \frac{\sum_{i_1,\ldots,i_k} f_{(i_1,\ldots,i_k)}}{\ln 2},$$

2. `CountMore`: The number of values t such that

$$\max_{i_1,\ldots,i_k} (\boldsymbol{e}^{(t)}_{(i_1,\ldots,i_k)})^{-1} f_{(i_1,\ldots,i_k)} \leq (1-\varepsilon) \frac{\sum_{i_1,\ldots,i_k} f_{(i_1,\ldots,i_k)}}{\ln 2},$$

3. `CountHeavy`: The number of values t such that

$$\sum_{(i_1,\ldots,i_k)} (\boldsymbol{e}^{(t)}_{(i_1,\ldots,i_k)})^{-1} f_{(i_1,\ldots,i_k)} \leq (C \log^2 n) \max_{i_1,\ldots,i_k} (\boldsymbol{e}^{(t)}_{(i_1,\ldots,i_k)})^{-1} f_{i_1,\ldots,i_k}.$$

Note that the algorithm can keep track of all these random variables only using $O(b)$ bits of space as follows: When processing the first set of discrete exponential random variables, the algorithm keeps track of cumulative sum and cumulative max corresponding to those set of random variables and at the end updates the counts appropriately. It discards the stored cumulative sum and cumulative max values and starts processing the second set of random variables and so on.

We now note using the properties of continuous exponential random variables that when the discrete exponential random variables are sampled in the above defined manner using a fully random string, then with probability $\geq 99/100$, using the union bound over the properties of continuous exponential random variables, the following happen:

1. `CountLess` $< m/2$,
2. `CountMore` $< m/2$, and
3. `CountHeavy` $= m$.

The first two properties from the fact that the median of $O(1/\varepsilon^2)$ independent copies of the random variable $(\sum_{(i_1,\ldots,i_k)} f_{(i_1,\ldots,i_k)})/\boldsymbol{e}$ concentrates in the interval $[(1-\varepsilon)\sum_{(i_1,\ldots,i_k)} f_{(i_1,\ldots,i_k)}/\ln 2, (1+\varepsilon)\sum_{(i_1,\ldots,i_k)} f_{(i_1,\ldots,i_k)}/\ln 2]$ and hence both the counts are at most $m/2$. The third property follows from using a union bound on the event in Lemma 13.2.1. Since the algorithm **Alg** uses only a space of $O(b)$ bits and the number of required random bits is $\exp(b)$, if the discrete exponential random variables are constructed using a pseudorandom string drawn from Nisan's PRG with a seed length of $O(b^2)$ bits, the above properties continue to hold with probability $\geq 98/100$. Hence, the protocol run with discrete exponential random variables constructed using Nisan's PRG continues to succeed in outputting a $1 \pm \varepsilon$ approximation to the higher-order correlation defined by the functions f and g also succeeds with probability $\geq 98/100$.