Taming Big Data By Streaming

by

Lin Yang

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

September, 2017

Abstract

Data streams have emerged as a natural computational model for numerous applications of big data processing. In this model, algorithms are assumed to have access to a limited amount of memory and can only make a single pass (or a few passes) over the data, but need to produce sufficiently accurate answers for some objective functions on the dataset. This model captures various real world applications and stimulates new scalable tools for solving important problems in the big data era.

This dissertation focuses on the following two aspects of the streaming model.

1. *Understanding the capability of the streaming model.* For a vector aggregation stream, i.e., when the stream is a sequence of updates to an underlying n-dimensional vector v (for very large n), we establish nearly tight space bounds on streaming algorithms of approximating functions of the form $\sum_{i=1}^{n} g(v_i)$ for nearly all functions g of one-variable and $l(v)$ for all symmetric norms l. These results provide a deeper understanding of the streaming computation model.

2. *Tighter upper bounds.* We provide better streaming k-median clustering algorithms in a dynamic points stream, i.e., a stream of insertion and deletion of points on a discrete Euclidean space ($[\Delta]^d$ for sufficiently large Δ and d). Our algorithms use $k \cdot \mathrm{poly}(d \log \Delta)$ space/update time and maintain with high probability an approximate k-median solution to the streaming dataset. All previous algorithms for computing an approximation for the k-median problem over dynamic data streams required space and update time exponential in d.

Acknowledgments

Foremost, I am sincerely grateful to my adviser Professor Vladimir (Vova) Braverman. I was enrolled at Johns Hopkins University as a Ph.D. student in physics. I was extremely lucky to meet Vova, who was so open-minded to accept me to work with him. Without Vova, I would not have been able to identify my interest and gift in theoretical computer science. In the years I was working in computer science, Vova supported me in every way he could. His guidance on how to do research in theoretical computer science has greatly benefited me in coming up solutions for hard problems. He helped me find excellent collaborators and helped me to advertise my research progress to top institutes such as MIT and UC Berkeley. His generous financial support has allowed me to focus on my research without a distraction.

I would also like to thank my co-advisor Professor Alex Szalay, who is also my advisor in Physics. Alex encouraged me to pursue my degree in Computer Science and helped me stay in both programs simultaneously.

I thank my collaborators, Professor Avrim Blum, Dr. Steve Chestnut, Professor

ACKNOWLEDGMENTS

Robert Krauthgamer, Professor Carey Priebe, Professor Christian Sohler, Professor Vinodchandran N. Variyam, Professor Mengdi Wang, Professor David Woodruff, Professor Tuo Zhao, Nikita Ivkin, Ananya Kumar, Harry Lang, Xingguo Li, Zaoxin Liu, Zhao Song, Zeyu Zhang, and Peilin Zhong. I am grateful for all the help, guidance and insightful discussions. I am thankful for Professor Tuo Zhao and Professor Mengdi Wang for helping me find great opportunities for the future.

I would like to express my gratitude to my dissertation committee, Professor Amitabh Basu, Professor Xin Li and Professor Carey Priebe, for their time and patience in examining my work. I am thankful to all the professors in the Computer Science department, especially Professor Scott Smith, who has always been kind and helpful to me since the day I started my master program in the department.

I am thankful for the administrative staffs in the department, Zack Burwell, Debbie Deford, Tonette McClamy and others. They are extremely responsive and have fulfilled all my requests with highly satisfactory services. My particular thanks are due to Zack and the former Academic Program Administrator Cathy Thornton, who made a great effort to make the dual Ph.D. degrees possible.

I have spent six memorable years at Johns Hopkins University with my friends, Steve Chestnut, Liang Dai, Lei Feng, Zaoxin Liu, Can You, Tuo Zhao, Yaofu Zhou, and many others. I cannot imagine how I could survive without the companion of these friends.

ACKNOWLEDGMENTS

I owe a great deal to my parents for their understanding and selfless support from the other side of the Earth. I have never faced the type of financial crisis that is usually faced by people with similar family backgrounds. They are supporting me at the very roots of my life – no matter where I am going, I will never feel alone.

Last but not least, I would like to express gratitude to my beloved wife, Lizi Xie for her love, care, understanding and support. We have known each other since we were high school students. I cannot imagine how much she has sacrificed by leaving behind her colorful life in China to be together with me. We have shared together every moment of our lives. Words are not enough to express my gratitude for her unconditional love and support.

Dedication

To my mother and father, Yanzhen Sheng & Zhongfa Yang.

To my beloved wife, Lizi Xie.

Contents

List of Figures

Chapter 1

Introduction

Data streams have emerged as a natural computational model for numerous applications of big data processing. In this model, algorithms are assumed to have access to a limited amount of memory and can only make a single pass (or a few passes) over the data, but need to produce sufficiently accurate answers for some objective statistics of the dataset. This model captures various real world applications and stimulates new scalable tools for solving important problems in the big data era. Typical examples of data streams include network traffic data,[1,2] data base transactions,[3–5] sensor network data flows[6] and satellite data.[7] Streaming algorithms also provide indispensable toolkits to different areas of computing, which include but are not limited to, computational astronomy and physics,[8] bioinformatics,[9,10] finance,[11] machine learning[12] and optimization.[13]

Since its first systematic study in a seminal paper by Alon, Matias and Szegedy (1996),[14] streaming algorithms have inspired numerous variants. Some well-studied models follow

below:

- *Aggregation Stream* In this model, an underlying vector $v \in \mathbb{R}^n$ (v can also represent a matrix) is initialized to $\mathbf{0}$. The stream is a sequence of pairs $S = ((a_1, \delta_1), (a_2, \delta_2), \ldots, (a_i, \delta_i))$, where each $a_i \in [n]$ and $\delta_i \in \mathbb{R}$ represents an increment to the a_i-th coordinate to v. For instance, at time t, the vector gains an update $v_{a_t} \leftarrow v_{a_t} + \delta_t$. If we consider the case δ_i as being restricted to positive integers, we obtain the *insertion-only stream* model. For general $\delta_i \in \mathbb{R}$, the model is also called the *turnstile model*. A typical task in the vector aggregation stream is as follows. Given a function $f : \mathbb{R}^n \to \mathbb{R}$, obtain an estimation to $f(v)$ at the end of the stream while making a single pass over the stream. The goal is to minimize the storage of the algorithm. A naïve solution to this problem is to store the entire vector v. However, such an algorithm is not practical since it uses space of at least $\Omega(n)$.

  The most well-studied functions in aggregation streams are ℓ_p norms. A long line of research has been devoted to investigating the space complexity of approximating ℓ_p norms[1], see e.g.[7,14–16] Here $\ell_p : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\ell_p(v) = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p}.$$

  In particular, Indyk & Woodruff[15] obtain the first nearly space optimal randomized algorithm for approximating ℓ_p norm by storing $\widetilde{\mathcal{O}}(n^{1-2/p})$ bits summary of the stream; Indyk[16] obtains the first nearly space optimal randomized algorithm for approximat-

[1] For $0 \leq p < 1$, ℓ_p is not a norm but still well-defined.

ing ℓ_p for $0 \leq p < 2$ by storing $\operatorname{poly}\log n$ bits summary of the stream; BJKS[17] obtain the first nearly optimal lower bound for ℓ_p.

For other functions, Braverman and Ostrovsky,[18] Braverman and Chestnut[19] and BCWY[20] investigate the family of G-sum functions, i.e.,

$$G(v) := \sum_i g(v_i),$$

for some function $g : \mathbb{R} \to \mathbb{R}$. The space complexity of streaming the family of G-sum functions are nearly completely characterized by the above papers. Recently, BBCKY[21] obtain the nearly tight space complexity results for symmetric norms. A symmetric norm $\ell : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is the set of norms that is invariant under coordinate permutation and sign flips of each coordinate. For instance,

$$\ell((v_1, v_2, \ldots, v_n)) = \ell((|v_{\sigma(1)}|, |v_{\sigma(1)}|, \ldots, |v_{\sigma(n)}|))$$

for arbitrary permutation $\sigma : [n] \to [n]$. BBCKY[21] show that the streaming complexity is $\widetilde{\mathcal{O}}(\operatorname{mmc}^2(\ell))$, where $\operatorname{mmc}(\cdot)$ is a function that characterizes the measure concentration behavior of ℓ (see Section 3 for more details).

- *Point Streams* In this case, we are observing a sequence of points $S = (p_1, p_2, \ldots, p_m)$ from the some space Ω. The goal is to obtain an estimation of some function f on the sequence S after one-pass over it. This class of streams also includes *graph streams*,[22] i.e., each point is an edge or vertex of the underlying graph. A typical example of f

is clustering and $\Omega = \mathbb{R}^d$ for some $d > 0$.[2] For example, in the k-clustering case, the function f is defined as

$$f(S) = \arg \min_{Z \subset \Omega : |Z| \leq k} \sum_{s \in S} \min_{z \in Z} \text{dist}^i(s, z).$$

for some distance function $\text{dist}(\cdot, \cdot)$ and $i \in \{1, 2\}$. For $i = 1$, the problem is called *k-median*. For $i = 2$, the problem is called *k-means*. There is a long line of research of k-clustering in the context of *insertion-only points stream* (the points already in the stream are not allowed to be removed). For example, Ref.[23–30] and many others have developed and improved streaming algorithms for computing a solution for k-means and k-median approximately. In the *dynamic points stream model*, the points are allowed to be deleted. In recent years, there has been a great deal of interest in dynamic streaming algorithms, e.g., Ref.[31–48] In 2004, Indyk[49] introduced the model for dynamic geometric data streams, in which a set of geometric points from a d-dimensional discrete space $\Omega = [\Delta]^d$ are updated dynamically, for some large Δ, i.e., the data stream is of the form insert(p) and delete(p) for $p \in [\Delta]^d$. Frahling and Sohler[50] develop the first streaming $(1 + \epsilon)$-approximation algorithms for k-means, k-median, as well as other geometric problems over dynamic data streams. In their paper, they propose an algorithm to maintain a *coreset* of size $O(k\epsilon^{-O(d)})$ for obtaining a $(1 \pm \epsilon)$ approximation of k-clustering including k-means and k-median. A coreset for clustering is a smaller weighted point set that summarizes the original geometric dataset. Solving the clustering problem over the coreset provides an approximate

[2] In the actual computation, Ω is discretized. Therefore the storage of each point costs a finite number of bits.

solution for the original problem. It is one of the fundamental tools for solving the k-clustering problem.

- *Stochastic Stream* In the stochastic stream, the updates of the stream are drawn from a distribution. The stochastic stream is a special case of the points stream and the aggregation stream except that we also require the algorithm to use a small number of data points. Since the stochastic stream has gained tremendous interest in the community of machine learning and optimization, we list it as a separate streaming model. Usual examples of stochastic streams are vector streams. For instance, $S = (v_1, v_2, \ldots, v_m)$, where each $v_i \in \mathbb{R}^d$ is a d-dimensional random vector drawn independently from a multi-variate normal distribution with a fixed covariance matrix. If we fix the distribution to be arbitrary deterministic distributions, the model reduces to the usual insertion-only points stream. Recently, there has been a great deal of interest in designing algorithm for problems on stochastic points streams. For instance, in streaming principle component analysis (PCA), the goal is to obtain the leading-k eigenvectors of the covariance matrix while using a small amount of space and a small number of streaming points. The state-of-art algorithms are the Oja's algorithm and its variants.[51–53] Some other examples include streaming singular value decomposition (SVD)[54] and streaming independent component analysis (ICA).[55]

Our Contributions Generally speaking, the goal of a streaming algorithm is to *compute a function* of the input data *efficiently* in both space and time. Most of the work over the last decade has been devoted to designing algorithms for different but specific functions.

However, there is a lack of generic characterization, which diminishes both the theoretical and the practical values of these streaming algorithms. For example, a theory may show that some particular function is easy to be approximated; however, the function might be slightly changed in practice due to implementation concerns, and then immediately the theory ceases to hold. Moreover, a large number of important problems have polynomial or even exponential gaps over the problem scale between space/time upper and lower bounds.

To resolve these important issues in streaming algorithms, we focus on the following two aspects of the streaming model:

1. *Fundamental hardness.* My coauthors and I have established a hardness characterization for much broader classes of functions in the streaming model.[20,21] These results provide a deeper understanding of how streaming algorithms work. We present these results in detail in Chapter 2 and Chapter 3.

2. *Tighter upper bounds.* My coauthors and I have developed new algorithms for important problems with tighter upper bounds.[20,21,56,57] The algorithms enable important problems to be solvable in the big data era. In this dissertation, we present one of these results: namely a better streaming clustering algorithm in dynamic points stream. This result is presented in Chapter 4.

In the next few paragraphs, we elaborate the contributions in more detail.

Characterize broader classes of functions that admit efficient streaming algorithms. It is a fundamental question to ask which function class admits efficient streaming

and which does not. The first open problem of this kind was proposed by Alon, Matias, and Szegedy (1999) on the *g-sum* function over vector aggregation streams. Let $v \in \mathbb{R}^n$ be an n-dimensional vector, initialized as $\mathbf{0} \in \mathbb{R}^n$. The stream is composed of a sequence of updates to the coordinates of v (i.e., $+1, -1$s to each coordinate). A g-sum function is defined as $\sum_{i=1}^{n} g(v_i)$, where $g : \mathbb{R} \to \mathbb{R}$ is a real function. Many important applications can be formalized as estimating a g-sum function, e.g., network traffic monitoring.[2] About a decade of theoretical research has been devoted to the streaming space/time complexity analysis of g-sum estimation. However, these theoretical results were only on non-decreasing, periodic functions or other simple forms, for example, polynomials. In Chapter 2, we show that except for a very small class of functions, we give a complete answer of whether g-sum can be solved efficiently (in space) in the streaming model. In particular, for those functions for which g-sums admit efficient streaming, we develop space-efficient algorithms for them. The functions we are unable to characterize have provable exotic mathematical structures and have little practical value. This result nearly closes the open question of Alon, Matias, and Szegedy.

Along the way, we have focused on understanding the streaming space complexities of generic norms in a vector-updates stream. In Chapter 3, we bridge between the mathematical theory of finite normed space and the space complexity of streaming algorithms. We show that for streaming algorithms of symmetric norm $\ell : \mathbb{R}^n \to \mathbb{R}$ (a norm that is invariant under coordinate permutation and negation), the space complexity is $\widetilde{\Theta}(\mathrm{mmc}(\ell)^2)$. Here $\mathrm{mmc}(\ell)$ is called *modulus of concentration* and is (informally) defined as the ratio of the maximum and median of $\ell(X)$, where X is uniformly drawn from the n dimensional Euclidean

unit sphere. $\mathrm{mmc}(\ell)$ is an important quantity in the theory of finite-dimensional normed space that governs many phenomena in high-dimensional spaces, such as large-deviation bounds and the critical dimension in Dvoretzky's Theorem. We give nearly *optimal* algorithms and lower bounds for the streaming algorithms of every symmetric norm. This set of norms not only contains the existing results (e.g. for ℓ_p norms, $\widetilde{\Theta}(\mathrm{mmc}(\ell_p)^2)$ is the correct space-bound) but also includes many more important functions for different tasks (e.g. norms used in machine learning). To my best knowledge, our algorithms are the *first* nearly space optimal streaming algorithms for a large portion of important functions falling in our function class.

First polynomial space k-median algorithm for dynamic points stream. In Chapter 4, we present data streaming algorithms for the k-median problem in high-dimensional dynamic geometric points streams, i.e. streams allowing both insertions and deletions of points from a discrete Euclidean space $\{1, 2, \ldots \Delta\}^d$. Our algorithms use $k\epsilon^{-2}\mathrm{poly}(d \log \Delta)$ space/time and maintain with high probability a small weighted set of points (a coreset) such that for every set of k centers the cost of the coreset $(1+\epsilon)$-approximates the cost of the streamed point set. We also provide algorithms that guarantee only positive weights in the coreset with additional logarithmic factors in the space and time complexities. We can use this positively-weighted coreset to compute a $(1+\epsilon)$-approximation for the k-median problem by any efficient offline k-median algorithm. All previous algorithms for computing a $(1+\epsilon)$-approximation for the k-median problem over dynamic data streams required space and time exponential in d. Our algorithms can be generalized to metric spaces of bounded doubling dimension.

Chapter 2

Streaming Functions of One Variable on Frequency Vectors

This chapter is based on paper BCWY (PODS 2016).[20]

2.1 Background

One of the main open problems in the theory of data stream computation is to characterize functions of a frequency vector that can be computed, or approximated, efficiently. Here we characterize nearly all functions of the form $\sum_{i=1}^{n} g(|v_i|)$, where $v = (v_1, \ldots, v_n) \in \mathbb{Z}^n$ is the frequency vector of the stream. This is a generalization of the famous frequency moments problem, where $g(x) = x^k$ for some $k \geq 0$, described by Alon, Matias, and Szegedy,[14] who also asked:

CHAPTER 2. STREAMING FUNCTIONS OF ONE VARIABLE ON FREQUENCY VECTORS

"It would be interesting to determine or estimate the space complexity of the approximation of $\sum_{i=1}^{n} v_i^k$ for non-integral values of k for $k < 2$, or the space complexity of estimating other functions of the numbers v_i."

The first question was answered by Indyk and Woodruff[15] and Indyk[16] who were the first to determine, up to polylogarithmic factors, the space complexity of the frequency moments for all $k > 0$. We address the second question.

Braverman and Ostrovsky[18] and Braverman and Chestnut,[19] answer the second question for sums of the form given above when g is monotone. We extend their characterizations to nearly all nonmonotone functions of one variable. Specifically, we characterize the set of functions g for which there exists a sub-polynomial $(1 \pm \epsilon)$-approximation algorithm for the sum above. Our results can be adapted to characterize the set of functions approximible in poly-logarithmic, rather than sub-polynomial, space. Among the main techniques we use is the layering method developed by Indyk and Woodruff[15] for approximating the frequency moments, and one may view our results as an exploration of the power of this technique.

Our results also partially answer an open question of Guha and Indyk at the IIT Kanpur workshop in 2006[58] about characterizing sketchable distances - our results characterize nearly all distances that can be expressed in the form $d(u, v) = \sum_i g(|u_i - v_i|)$. Besides the aforementioned work on the frequency moments and monotone functions, other past work on this question was done by Guha, Indyk, and McGregor[59] as well as Andoni, Krauthgamer, and Razenshteyn.[60] Both of those papers address the same problem, but in domains different from ours. In the realm of general streaming algorithms, a result of Li, Nguyen, and Woodruff[61] shows that for any function g with a $(1 \pm \epsilon)$-approximation algorithm im-

plementable in sub-polynomial space, there exists a linear sketch that demonstrates its tractability. However, a drawback of that work is that the space complexity of maintaining the sketching matrix, together with computing the output, may be polynomially large. Our paper deals with a smaller class of functions than,[61] but we provide a zero-one law and explicit algorithms.

While several of our lower bounds can be shown via reduction from the standard index and set-disjointness communication problems, to prove lower bounds for the widest class of functions we develop a new class of communication problems and prove lower bounds for them, which may be of independent interest. The problem, which we call ShortLinearCombination, is: given a finite set $\{0, a_1, \ldots, a_r, b\}$ of possible frequencies, is there a frequency in the stream of value b? Perhaps surprisingly, the communication complexity of ShortLinearCombination depends on the magnitudes of the coefficients in any linear combination expressing b in terms of $a_1, \ldots, a_r$. We develop optimal communication bounds for this problem. This hints at the subtleties in proving lower bounds for a wide class of functions g, since by defining $g(b) \geq n \cdot \max_i g(a_i)$, the communication problem just described is a special case of characterizing the streaming complexity of all functions g. Our lower bounds provide a significant generalization of the set-disjointness problem, which has found many applications to data streams.

2.1.1 Applications

We now describe potential applications of this work.

Log-likelihood Approximation One use for the approximation of non-monotonic functions is the computation of log-likelihoods. Here, the coordinates of the streamed vector v are taken to be i.i.d. samples from a discrete probability distribution. The likelihood of the vector v, under a distribution with probability mass function $p(x)$, $x \in \mathbb{Z}$, is $L(v) = \prod_{i=1}^{n} p(v_i)$, and its log-likelihood is $\ell(v) = -\sum_{i=1}^{n} \log p(v_i)$. Notice that $\ell(v)$ is of the form $\sum_{i=1}^{n} g(v_i)$, for $g(x) = -\log p(x)$. When $p(\cdot;\theta)$ is the distribution of a non-negative random variable or if $p(x;\theta)$ is symmetric about $x = 0$, our results can be applied to determine whether there is a streaming algorithm that efficiently approximates ℓ. In general, $-\log p(x)$ is not a monotonic function of x. For example, $p(x) = \lambda \frac{x^\alpha}{x!} e^{-\alpha} + (1-\lambda)\frac{x^\beta}{x!} e^{-\beta}$ is a mixture of two Poissons, which is generally not monotonic. For any constants $\lambda, \alpha, \beta > 0$, the Poisson mixture log-likelihood $-\log p(x)$ satisfies our three criteria (to be described later), hence the log-likelihood of the stream can be approximated in poly-logarithmic space. Continuous distributions can be handled similarly by discretization.

Suppose that the distribution p comes from a family of probability distributions $p(\cdot;\theta)$ parameterized by $\theta \in \Theta$. When an efficient approximation algorithm exists, we describe a linear sketch from which a $(1 \pm \epsilon)$-approximation $\widehat{\ell}$ to $\ell(v)$ can be extracted with probability at least $2/3$. The form of the sketch is independent of the function g, so if $-\log p(x;\theta)$ satisfies our conditions for approximability for all $\theta \in \Theta$, we derive an approximation to each of the log-likelihoods $\ell(v;\theta)$ that are separately correct with probability $2/3$ each. If Θ is a discrete set then this allows us to find an approximate maximum likelihood estimator for θ with only an $O(\log|\Theta|)$ factor increase in storage. Recall, the maximum likelihood estimator is $\widehat{\theta} = \operatorname{argmin}_{\theta\in\Theta} \ell(\theta, v)$. Indeed, as usual we can drive the error probability

down to any $\delta > 0$ with $O(\log 1/\delta)$ repetitions of the algorithm, so an additional factor of $O(\log |\Theta|)$ in the space complexity is enough. If $\widehat{\ell}(\theta; v)$ denote our approximations, then $\widehat{t} = \text{argmin}_{\theta \in \Theta} \widehat{\ell}(\theta; v)$ is the approximate maximum likelihood estimate, which has the guarantee that $\ell(\widehat{t}; v) \leq (1 + \epsilon)\ell(\widehat{\theta}; v)$. When Θ is not finite one may still be able to apply our results by discretizing Θ with poly(n) points and proceeding as above; whether this works depends on the behavior of $p(x; \theta)$ as a function of θ.

Utility Aggregates Consider an online advertising service that charges its customers based on the number of clicks on the ad by web users. More clicks should result in a higher fee, but an exceptionally high number of clicks from one user make it likely that he is a bot or spammer. Customers may demand that the service discount the cost of these spam clicks, which means that the fee is a non-monotonic function of the number of clicks from each user.

The ads application is an example of a non-monotonic utility function. Many utility functions are naturally non-monotonic, and they can appear in streaming settings. For example, extremely low or high trading volume in a stock indicates abnormal market conditions. A low volume of network traffic might be a sign of damaged or malfunctioning equipment whereas a high volume of network traffic is one indicator of a denial-of-service attack.

Database Query Optimization A prominent application for streaming algorithms, indeed one of the initial motivations,[14] is to database query optimization. A query optimizer

is a piece of software that uses heuristics and estimates query costs in order to intelligently organize the processing of a database query with the goal of reducing resource loads (time, space, network traffic, etc.). Scalable query optimizers are needed for scalable database processing,[62,63] and poor query efficiency has been cited as one of the main drawbacks of existing implementations of the MapReduce framework.[64] Streaming algorithms and sketches are a natural choice for such heuristics because of their low computational overhead and amenability to distributed processing. By expanding the character of statistics that can be computed efficiently over data streams, our results could allow database query optimizers more expressive power when creating heuristics and estimating query costs.

Encoding Higher Order Functions An obvious generalization of the problem we consider is to replace the frequency vector f_i, $i \in [n]$, with a frequency matrix f_{ij}, $i \in [n]$ and $j \in [k]$, and ask whether there is a space efficient streaming algorithm that approximates $\sum_{i=1}^{n} g(f_{i,1}, \ldots, f_{i,k})$. These could, for example, allow one to express more complicated database queries which first filter records based on one attribute and then sum up the function values applied to another attribute on the remaining records.

Let us demonstrate that when the coordinates of f are bounded and k is not too large, we can replace this sum with a function of a single variable. Suppose that $0 \leq f_{ij} \leq b-1 \in \mathbb{N}$, for all i, j. Upon receiving an update to coordinate (i, j) we replace it with b^j copies of i. The new frequencies are polynomially bounded if $b^k = \text{poly}(n)$.

Call the new frequency vector $f' \in \mathbb{Z}^n$. The sequence of values $f_{i1}, \ldots, f_{ik}$ is immediately available as the base-b expansion of the number f'_i, so we are able to write

$g'(f'_i) = g(f_{i1}, \ldots, f_{ik})$, where g' first recovers the base-b expansion and then applies g. Our desire is to approximate $\sum_i g'(f'_i)$. Given even a well behaved function g, it is very unlikely that g' will be monotone, hence the need for an approximation algorithm for non-monotonic functions if this approach is taken. It is also very likely that, because of the construction, g' has high local variability. This is a significant problem for algorithms that use only one pass over the stream (as we show by way of a lower bound), but we also present a two pass algorithm that is not sensitive to local variability.

2.1.2 Problem Definition

A stream of length m with domain $[n]$ is a list $D = \langle (i_1, \delta_1), (i_2, \delta_2), \ldots (i_m, \delta_m) \rangle$, where $i_j \in [n]$ and $\delta_j \in \mathbb{Z}$ for every $j \in [m]$. The *frequency vector* of a stream D is the vector $V(D) \in \mathbb{Z}^n$ with i^{th} coordinate $v_i = \sum_{j:i_j=i} \delta_j$. A processor can read the stream some number $p \geq 1$ times, in the order in which it is given, and is asked to compute a function of the stream. We allow the processor to use randomness and we only require that its output is correct with probability at least $2/3$.

Our algorithms are designed for the *turnstile* streaming model. In particular, there exists $M \in \mathbb{N}$ and it is promised that $V(D) \in \{-M, -M+1, \ldots, M\}^n$ and the same holds for any prefix of the list D. Our lower bounds, on the other hand, hold in the more restrictive *insertion-only* model which has $\delta_j = 1$, for all j. We use $\mathcal{D}(n, m)$ to denote the set of all turnstile streams with domain $[n]$ and length at most m.

CHAPTER 2. STREAMING FUNCTIONS OF ONE VARIABLE ON FREQUENCY VECTORS

Given a function $g : \mathbb{R} \to \mathbb{R}$ and vector $V = (v_1, v_2, \ldots, v_n)$, let

$$g(V) := \sum_{i=1}^{n} g(|v_i|).$$

Definition 2.1.1. Given $g : \mathbb{Z}_{\geq 0} \to \mathbb{R}$, $\epsilon > 0$, and $D \in \mathcal{D}(n, m)$ the problem of (g, ϵ)-SUM on stream D is to output an estimate $\widehat{G}$ of $g(V(D))$ such that

$$P\left((1-\epsilon)g(V(D)) \leq \widehat{G} \leq (1+\epsilon)g(V(D))\right) \geq 2/3.$$

We often omit ϵ and refer simply to g-SUM when the value of ϵ is clear from the context.

The choice of 2/3 here is arbitrary, since given a g-SUM algorithm the success probability can be improved to $1 - 1/\text{poly}(n)$ by repeating it $O(\log n)$ times in parallel and taking the median outcome.

Our goal is to classify the space complexity of g-SUM. We ask: for which functions g can (g, ϵ)-SUM be solved in space that is sub-polynomial in n? We call such functions *tractable*. An $\Omega(\epsilon^{-1})$ lower bound applies for nearly every function g, thus we only require that the algorithm solve (g, ϵ)-SUM whenever ϵ decreases sub-polynomially. Our main results separately answer this question in the case in which the processor is allowed one pass over the stream and in the case in which $O(1)$ passes are allowed. We are able to classify the space complexity of almost all functions, though a small set of functions remains poorly understood.

2.1.3 Organization

Section 2.2 gives a high-level description of our results and states our main theorems. We put off some of the more technical definitions until Section 2.3, which gives the definitions needed to precisely state our main results as well as background on the main techniques for our proofs. Next, Section 2.4 includes the intuition and proofs for our main theorems. It describes one and two-pass algorithms and lower bounds. Section 2.5 discusses the functions that we are unable to characterize in more depth.

2.2 Our Results

To begin with, we separate the class of functions $g : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ into two complementary classes: "normal functions" and "nearly periodic functions". In fact, we do this differently for 1-pass g-SUM than for multi-pass g-SUM. We define $\mathbb{S}$-normal functions, for studying one pass streaming space complexity, and $\mathbb{P}$-normal functions for multiple passes. The complements of these two sets are the $\mathbb{S}$-nearly periodic functions and $\mathbb{P}$-nearly periodic functions, respectively. The distinction between $\mathbb{S}$ and $\mathbb{P}$ is not important to understand our results at a high level, so we omit them for the rest of this section. See Section 2.5 for more about this class of functions. We postpone the technical definitions until Section 2.3, and give an informal overview here.

We are able to characterize the normal functions based on three properties which we call *slow-jumping*, *slow-dropping*, and *predictable*. Slow-jumping depends on the rate of

increase. Roughly, a function is slow-jumping if it doesn't grow much faster than $y = x^2$ at every scale. Slow-dropping governs the rate of decrease of a function. A function is slow dropping if decreases no faster than sub-polynomially at every point. Finally, if a function is predictable then it satisfies a particular tradeoff between its growth rate and local variability. Interestingly, not just the properties themselves are important for our proofs, but so is the interplay between them. Precise definitions for these properties are given in Section 2.3.

Our proofs are by finding heavy hitters, for the upper bound, and by reductions from communication complexity for the lower bound. See Section 2.4.1 for more thorough discussion of how the three properties relate to heavy-hitters and the other techniques we use.

Zero-One Laws for Normal Functions

A nonnegative function f is called a sub-polynomial function if, for all $\alpha > 0$, $\lim_{x\to\infty} x^\alpha f(x) = \infty$ and $\lim_{x\to\infty} x^{-\alpha} f(x) = 0$. Formally, we study the class of functions g such that g-SUM can be solved with sub-polynomial accuracy $\epsilon = \epsilon(n)$ using only an amount of space which is sub-polynomial. We call such functions p-pass tractable if the algorithm uses p passes. In this paper we prove the following theorems.

Theorem 2.2.1 (1-pass Zero-One Law)**.** A function $g \in \mathcal{G}$ is 1-pass tractable and normal if and only if it is slow-jumping, slow-dropping, and predictable.

Theorem 2.2.2 (2-pass Zero-One Law)**.** A function $g \in \mathcal{G}$ is 2-pass tractable and normal if and only if it is slow-dropping and slow-jumping.

The main difference between the two theorems is that predictability is not needed in two passes. The message is that large local variability can rule out 1-pass approximation algorithms but not 2-pass approximation algorithms. Theorem 2.2.2 extends to any subpolynomial number of passes.

Most functions one encounters are normal, which include convex, concave, monotonic, polynomial, or trigonometric functions, as well as functions of regular variation and unbounded Lipschitz-continuous functions.

2.3 Preliminaries

In describing the requirements and space efficiency of our algorithms for g-SUM, we use the set of sub-polynomial functions.

Definition 2.3.1. A function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is *sub-polynomial* if for any $\alpha > 0$, the following two conditions are satisfied: a) $\lim_{x\to\infty} x^{\alpha} f(x) = \infty$ and b) $\lim_{x\to\infty} x^{-\alpha} f(x) = 0$.

We use subpoly(x) to represent the set of sub-polynomial functions in the variable x. Examples of sub-polynomial functions include polylogarithmic functions and some with faster growth, like $2^{\sqrt{\log n}}$. The set of poly-logarithmic functions is a subset of the sub-polynomial functions. Formally, we study the class of functions g such that (g, ϵ)-SUM can be solved with any accuracy $\epsilon \in$ subpoly(n), using only sub-polynomial space. Our algorithms assume an oracle for computing g and that the storage required for the value $g(x)$ is sub-polynomial in x. Regardless, our sketches are sub-polynomial in size, but without these assumptions it could take more than sub-polynomial space to compute the approx-

imation from the sketch. We restrict ourselves to the case $M \in \text{poly}(n)$. This restriction is reasonable because if it grows, say, exponentially in n, then an algorithm can store the entire frequency vector and compute $g(V(D))$ exactly in polylog(nM) space.

Definition 2.3.2. A function $g : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ is *p-pass tractable* if given any sub-polynomial function $h(x)$ and any $\epsilon \geq 1/h(nM)$ there exists a sub-polynomial function $h^*(x)$ and a p-pass algorithm $\mathcal{A}$ that solves (g, ϵ)-SUM for all streams in $\mathcal{D}(n, m)$ and $n, M \geq 1$ using no more than $h^*(nM)$ space in the worst case.

Replacing the storage requirement, which is $h^*(nM)$, with $h^*(n) \log M$ would also be natural, but since we consider $M \in \text{poly}(n)$ the two definitions are equivalent. It simplifies our notation a little bit to write $h^*(nM)$. Unless otherwise noted, for the rest of this paper we require that $g(0) = 0$ and $g(x) > 0$, for all $x > 0$. The choice $g(0) = 0$ is equivalent to requiring that $g(V(D))$ does not depend on the particular choice of n in the model; specifically it avoids the following behavior: if $g(0) \neq 0$ then given a single stream D the value of $g(V(D))$ differs depending on the choice of the dimension, even though the stream remains the same. Functions with $g(0) \neq 0$ may be of interest for some applications. The laws for these functions are very similar to the case when $g(0) = 0$ and we provide them in Appendix 2.6.1.

As shown in,[65] functions with $g(x) = 0$, for some integer $x > 0$, are not n^α-pass tractable for any $\alpha < 1$, unless g is nonnegative and periodic with period $\min\{x > 0 \mid g(x) = 0\}$, which must exist if g is periodic and $g(0) = 0$. It is also shown in[65] that if a non-linear function g takes both positive and negative values, then g is not n^α-pass tractable, for any $\alpha < 1$. Without loss of generality, we can assume $g(1) = 1$ because a multiplicative approx-

imation algorithm for the function $g'(x) := g(x)/g(1)$ is also a multiplicative approximation algorithm for g. Finally, for simplicity of notation we will often extend the domain of g symmetrically to $\mathbb{Z}$, i.e., setting $g(x) = g(-x)$, for all $x \in \mathbb{Z}_{\geq 0}$, which allows us the simpler notation $g(|v_i|) = g(v_i)$. In summary, we study the functions in the class

$$\mathcal{G} \equiv \{g : \mathbb{Z}_{\geq 0} \to \mathbb{R}, g(0) = 0, g(1) = 1, \forall x > 0, g(x) > 0\}$$

Our results are based on three characterizations of the variation of the functions. We now state the technical definitions mentioned in Section 2.2.

Definition 2.3.3. A function $g \in \mathcal{G}$ is *slow-jumping* if for any $\alpha > 0$, there exists $N > 0$, such that for any $x < y$ if $y \geq N$, then $g(y) \leq \lfloor \frac{y}{x} \rfloor^{2+\alpha} x^{\alpha} g(x)$.

Slow-jumping is a characterization of the rate of growth of a function. Examples of slow-jumping functions are x^p, for $p \leq 2$, $x^2 2^{\sqrt{\log x}}$, and $(2 + \sin x)\, x^2$, while functions like 2^x and x^p, for any $p > 2$, are not slow jumping because they grow too quickly.

Definition 2.3.4. A function $g \in \mathcal{G}$ is *slow-dropping* if for any $\alpha > 0$ there exists $N > 0$, such that for any $x < y$, if $y \geq N$, then $g(y) \geq g(x)/y^{\alpha}$.

Whether a function is slow-dropping is determined by its rate of decrease. The functions[1] $(\log_2 1 + x)^{-1} \mathbf{1}(x > 0)$ and $(2 + \sin x)\, x^2$ are slow-dropping, but any function with polynomial decay, i.e. x^{-p}, for $p > 0$, is not slow-dropping.

Given a function g, $x \in \mathbb{N}$, and $\epsilon > 0$ define the set: $\delta_\epsilon(g, x) := \{y \in \mathbb{N} \mid |g(y) - g(x)| \leq \epsilon g(x)\}$.

[1] $\mathbf{1}(\cdot)$ denotes the indicator function.

Definition 2.3.5. A function g is *predictable* if for every $0 < \gamma < 1$ and sub-polynomial ϵ, there exists N such that for all $x \geq N$ and $y \in [1, x^{1-\gamma})$ with $x + y \notin \delta_{\epsilon(x)}(g, x)$ we have $g(y) \geq x^{-\gamma} g(x)$.

Predictability is governed by the local variability of a function and its rate of growth. For example, the function $g(x) = x^2$ is predictable because $g(x+y)/g(x) = (1 + \frac{y}{x})^2 \approx 1$ when $y \ll x$, or more precisely $\pi_\epsilon(x) \supseteq \{y \mid |x - y| \leq \epsilon x/3\}$. That function has low local variability. On the other hand, the function $(2 + \sin x)\mathbf{1}(x > 0)$ is locally highly variable but still predictable. Notice that even $x + 1 \notin \pi_\epsilon(x)$, for say $\epsilon = 0.01$, but $g(1)/g(x) \geq 1/3$ so the inequality in the definition of predictability is satisfied. A negative example is the function $(2 + \sin x)x^2$, which is not predictable because it varies quickly (by a multiplicative factor of 3) and grows with x.

As we will show, the three conditions above can be used to characterize nearly every function in $\mathcal{G}$ in both the single-pass and $O(1)$-pass settings. We remark here that the characterization has already been completed for monotonic functions. The tractability condition for nondecreasing g proved by[18] is equivalent to g being slow-jumping and predictable. For nonincreasing functions, it is a consequence of[19] that polynomially decreasing functions are not tractable while sub-polynomially decreasing functions are tractable. The "nearly periodic" functions are formally defined next.

Let $\mathbb{S}$ be the set of non-increasing sub-polynomial functions on domain $\mathbb{Z}_{\geq 0}$ and $\mathbb{P}$ be the set of strictly increasing polynomial functions on $\mathbb{Z}_{\geq 0}$. We now define the sets of $\mathbb{S}$-nearly periodic functions and $\mathbb{P}$-nearly periodic functions. The motivation for these definitions

jumps ahead to our lower bounds, but we will explain it here. The reductions used in some of our lower bounds boil down to finding three integers $x < y$ and $x+y$ such that $g(x) \gg g(y)$ and $g(x) \not\approx g(x+y)$. The reductions fail when $g(x) \approx g(x+y)$, where the meaning of "$\approx$" depends on whether we are bounding 1-pass or multi-pass algorithms. For the 1-pass reduction to fail it is necessary that $\frac{1}{g(x)}|g(x) - g(x+y)|$ decreases as x increases, hence the $\mathbb{S}$-nearly periodic functions. The 2-pass reduction fails as long as $\max(\frac{g(x)}{g(x+y)}, \frac{g(x+y)}{g(x)})$ is not polynomially large in y, hence the $\mathbb{P}$-nearly periodic functions.

Definition 2.3.6. Given a set of functions $\mathcal{S}$, call $g(x)$ *$\mathcal{S}$-nearly periodic*, if the following two conditions are satisfied.

1. There exists $\alpha > 0$ such that for any constant $N > 0$ there exists $x, y \in \mathbb{N}$, $x < y$ and $y \geq N$ such that $g(y) \leq g(x)/y^{\alpha}$. Call such a y an *α-period of g*;

2. For any $\alpha > 0$ and any error function $h \in \mathcal{S}$ there exists $N_1 > 0$ such that for all α-periods $y \geq N_1$ and all $x < y$ such that $g(y)y^{\alpha} \leq g(x)$, we have $|g(x+y) - g(x)| \leq \min\{g(x), g(x+y)\}h(y)$.

A function g is *$\mathcal{S}$-normal* if it is not $\mathcal{S}$-nearly periodic.

The first condition states that the function is not slow-dropping. For example, if g is bounded then there is an increasing subsequence along which g converges to 0 polynomially fast. To understand the second condition, take h to be a decreasing function or a small constant. In loose terms, it states that if $x < y$ and $g(x) \gg g(y)$, then $g(x) \approx g(x+y)$. The choice of the set of functions $\mathcal{S}$ determines the relative accuracy implied by "$\approx$". We will apply the definition with $\mathcal{S}$ set to either $\mathbb{S}$ or $\mathbb{P}$, that is, with either subpolynomially or

polynomially small relative accuracy.

An $\mathbb{S}$-nearly periodic function (it is also $\mathbb{P}$-nearly periodic) can be constructed as follows. For each $x \in \mathbb{N}$, let $i_x = \max\{j : 2^j | x\}$ and let $g_{np}(0) = 0$ and $g_{np}(x) = 2^{-i_x}$. For example, $g_{np}(1) = 1$, $g_{np}(2) = 1/2$, $g_{np}(3) = 1$, $g_{np}(4) = 1/4$, etc. The proof that g is nearly periodic appears in Section 2.5, but for now notice that it satisfies the first part of the definition because $g_{np}(2^k) = 2^{-k}$ and, as an example of the second part, we have $g_{np}(2^k + 1) = g_{np}(1) = 1$. One may guess that a streaming algorithm for such an erratic function requires a lot of storage. In fact, g_{np} can be approximated in polylogarithmic space! Section 2.5 has the proof.

The following containment follows directly from the definition of the nearly periodic functions.

Proposition 2.3.7. Every $\mathbb{S}$-nearly periodic function is also $\mathbb{P}$-nearly periodic. Every $\mathbb{P}$-normal function is $\mathbb{S}$-normal.

2.3.1 Heavy Hitters, Communication Complexity, and CountSketch

Our sub-polynomial space algorithm is based on the Recursive Sketch of Braverman and Ostrovsky,[66] which reduces the problem to that of finding heavy hitters.

Definition 2.3.8. Given a stream $D \in \mathcal{D}(n, m)$ and a function g, call $j \in [n]$ a (g, λ)-*heavy hitter* of $V(D)$ if $g(|v_j|) \geq \lambda \sum_{i \neq j} g(|v_i|)$. We will use the terminology λ-heavy hitter when g is clear from the context.

Given a heaviness parameter λ and approximation accuracy ϵ, a heavy hitters algorithm

returns a set $\{i_1, i_2, \ldots\}$ containing every (g, λ)-heavy hitter and also returns a $(1 \pm \epsilon)$-approximations to $g(v_{i_j})$ for each i_j in the set. Such a collection of pairs is called a (g, λ, ϵ)-cover.

Definition 2.3.9. Given a stream $D \in \mathcal{D}(n, m)$, a (g, λ, ϵ)-cover of $V(D)$ is a set of pairs $\{(i_1, w_1), (i_2, w_2), \ldots, (i_t, w_t)\}$, where $t \leq n$ and the following are true:

1. $\forall k \in [t]$, $|w_k - g(|v_{i_k}|)| \leq \epsilon g(|v_{i_k}|)$, and

2. if j is a (g, λ)-heavy hitter of $V(D)$, then there exists $k \in [t]$ such that $i_k = j$.

Formally, a p-pass $(g, \lambda, \epsilon, \delta)$-heavy hitter algorithm is a p-pass streaming algorithm that outputs a (g, λ, ϵ)-cover of $V(D)$ with probability at least $(1 - \delta)$.

Theorem 2.3.10 ([66])**.** Let $\lambda = \frac{\epsilon^2}{\log^3 n}$ and $\delta = \frac{1}{\log n}$. If there exists a $(g, \lambda, \epsilon, \delta)$-heavy hitter algorithm using s bits of space, then there exists a (g, ϵ)-SUM algorithm using $O(s \log n)$ bits of space.

A similar approach has been used in other streaming algorithms; it was pioneered by Indyk and Woodruff[15] for the problem of approximating the frequency moments. Their algorithm uses the streaming algorithm CountSketch of Charikar, Chen, and Farach-Colton,[67] and ours will as well. Given λ, ϵ, and δ, a CountSketch using space $O(\frac{1}{\lambda \epsilon^2} \log \frac{n}{\delta} \log M)$ gives an approximation $\widehat{v}_i$ to the frequency of every item $i \in [n]$ with the following guarantee: with probability at least $(1 - \delta)$, $|v_i - \widehat{v}_i| \leq \epsilon \sqrt{\lambda F_2}$ for all $i \in [n]$. By adjusting the parameters slightly we can treat the output of CountSketch$(\lambda, \epsilon, \delta)$ as a set of $k = O(1/\lambda)$ pairs $(i_j, \widehat{v}_{i_j})$ such that $\{i_j\}_{j=1}^k$ contains all of the λ-heavy hitters for F_2 and $|v_{i_j} - \widehat{v}_{i_j}| \leq \epsilon \left(\sum_{j>k} \bar{v}_j^2 \right)^{1/2} \leq \epsilon \sqrt{\lambda F_2}$, for all $j = 1, 2, \ldots, k$, where $(\bar{v}_1, \ldots, \bar{v}_n)$ is a reorder-

ing of the frequency vector from largest to smallest absolute magnitude.

We establish our lower bounds using communication and information complexity. Several of our lower bounds can be shown by reduction from standard INDEX and DISJ problems, as well as a combination of them (related combined problems were used in [65,68]). In the INDEX(n) problem, there are two players Alice, given a set $A \subseteq [n]$, and Bob, given $b \in [n]$. Alice sends a message to Bob, and with probability at least $2/3$, Bob must determine whether $b \in A$. Any randomized one-way protocol for INDEX(n) requires Alice to send $\Omega(n)$ bits.[69] In an instance of $\mathsf{DISJ}(n,t)$ there are t players each receiving a subset of $[n]$ with the promise that either the sets are pairwise disjoint, or there is a single item that every set contains and other than this single item, the sets are pairwise disjoint. The players must determine which case they are in. The randomized, unrestricted communication complexity of $\mathsf{DISJ}(n,t)$ is $\Omega(n/t)$.[17,70–72] In the $\mathsf{DISJ}{+}\mathsf{IND}(n,t)$ problem, $t+1$ players are given sets $A_1, A_2, \ldots, A_{t+1} \subseteq [n]$, such that $|A_{t+1}| = 1$, with the promise that either the sets are disjoint or there is a single element contained in every set and they are otherwise disjoint. A standard argument shows the one-way communication complexity of $\mathsf{DISJ}{+}\mathsf{IND}(n,t)$ is $\Omega(n/t(\log n))$ (see Theorem 2.6.12).

To obtain stronger lower bounds for the nearly periodic function sums, we use the information complexity framework.

Definition 2.3.11 (ShortLinearCombination problem)**.** Let $u = (u_1, u_2, \ldots u_r)$ be a vector in $\mathbb{Z}^r$ for some integer r. Let $d > 0$ be an integer not in $\{|u_1|, |u_2|, \ldots, |u_r|\}$. A stream S with frequency vector v is given to the player, v is promised to be from $V_0 = \{u_1, u_2, \ldots, u_r, 0\}^n$ or $V_1 = \{\mathrm{EMB}(v,i,e) \mid v \in V_0, i \in [n], e \in \{-d,d\}\}$, where $\mathrm{EMB}(v,i,e)$ is to embed e onto

the i-th coordinate. The (u, d)-DIST problem is to distinguish whether $v \in V_0$ or $v \in V_1$.

This problem is a generalization of the DISJ(n, t) problem. We show a $\Omega(n/q^2)$ bits lower bound for it where $q = \sum_{i=1}^{r} |q_i|$ and q_i are the integers such that $\min_q\{q_1, q_2, \ldots, q_r \mid \sum_{i=1}^{r} q_i u_i = d\}$. For more details, we refer the reader to Appendix 2.6.7.

2.4 Zero-One Laws for Normal Functions

First, we develop some intuition about why the three conditions, slow-dropping, slow-jumping, and predictable, characterize tractable normal functions. Next comes the two pass and one pass algorithms followed by the lower bounds. Finally, we prove Theorem 2.2.1 and 2.2.2 in Section 2.4.6.

2.4.1 Our Techniques

The upper and lower bounds are both established using heavy hitters. Using a $\frac{\epsilon^2}{\log^3 n}$-heavy hitters subroutine that identifies each heavy hitter H and also approximates $g(f_H)$, the algorithm of Braverman and Ostrovsky[66] (Theorem 2.3.10) solves g-SUM with $O(\log n)$ storage overhead. We will show that if g is tractable then a subpolynomially sized CountSketch suffices to find heavy hitters for g. On the lower bounds side, typical streaming lower bounds, going back to the work of Alon, Matias, and Szegedy, are derived from reductions to communication complexity that only require the streaming algorithm to detect the presence of a heavy hitter. Thus the problem of characterizing tractable functions can be reduced to characterizing the set of functions admitting subpolynomial heavy hitters algorithms.

Now let us explain how each of the three properties relates to heavy hitters. For this discussion let $\alpha = \epsilon^2/\log^3 n$ be the heaviness parameter needed by Braverman's and Ostrovsky's algorithm and let $H \in [n]$ be the identity of some α-heavy hitter.

Slow-jumping and Slow-dropping A function that satisfies these two conditions cannot increase much faster than a quadratic function nor can it decreasy rapidly. This means that if H is a heavy hitter for g then it is also a heavy hitter for F_2. Concretely, suppose that there exists an increasing subpolynomial function h such that $g(y)/g(x) \leq (y/x)^2 h(y)$ and $g(x) \geq g(y)/h(y)$, for all $x \leq y$, so that g is slow-jumping and slow-dropping (Propositions 2.4.1 and 2.4.2 show that such an h always exists if g is slow-jumping and slow-dropping). A bit of algebra shows that $g(f_H) \geq \alpha \sum_i g(f_i)$ implies $f_H^2 \geq \frac{\alpha}{h(M)^2} \sum_j f_j^2$, so H is a $\alpha' := \frac{\alpha}{h(M)^2}$-heavy hitter for F_2. CountSketch suffices to identify all α'-heavy hitters for F_2 with $O(\alpha' \log^2 n)$ bits, so this gives an α-heavy hitters algorithm for g that still uses subpolynomial space.

If a function is not slow-jumping then a heavy hitter may be too small to detect and we prove a lower bound by reduction from DISJ+IND in a very similar fashion to the original lower bound for approximating the frequency moments by Alon, Matias, and Szegedy.

If a function is not slow-dropping then f_H may be hidden below many small frequencies. In this case, a reduction from the communication complexity of INDEX usually works to establish a one-pass lower bound (two player DISJ can be used for a multipass lower bound). Here is a preview of the reduction that will also explain the genesis of the nearly periodic functions. Suppose $g(1) = 1$ and there is an increasing sequence $n_k \in \mathbb{N}$ such that

$g(n_k) \leq 1/n_k$. Alice and Bob can solve an instance $(A \subseteq [n_k], b \in [n_k])$ of INDEX as follows. Alice creates a stream where every item in A has frequency n_k and the items in $[n_k] \setminus A$ do not appear. To Alice's stream Bob adds one copy of his item and they jointly run a streaming algorithm to approximate $g(f)$. The result is an approximation to either $|A|g(n_k) + 1$ or $(|A| - 1)g(n_k) + g(n_k + 1)$. If $g(n_k + 1)$ differs significantly from $g(n_k) + 1 \approx 1$ then Bob can determine whether $b \in A$ by checking the approximation to $g(f)$. The reduction fails if $g(n_k + 1) \approx 1$ and such a fuction may be tractable! A prime example is g_{np} from Section 2.3, which we demonstrate in Section 2.5.

At this point, one might have in mind our two pass heavy hitters algorithm. The algorithm is to use a CountSketch on the first pass to identify a subpolynomial size set of items containing all α-heavy hitters and then use the second pass just to tabulate the frequency of each of those items exactly. The details and proof of correctness for that algorithm are in Section 2.4.2. Local variability of the function is irrelevant for two passes because we can compute the frequencies exactly during the second pass, and that is why a function need not satisfy predictability to be two pass tractable. A one pass algorithm will have to get by with approximate frequencies.

Predictable Predictability is a condition on the local variability of the function. It roughly says that if $y \ll x$ then either $g(x + y) \approx g(x)$ or $g(y)$ is on the same scale[2] as $g(x)$. The first conclusion has an obvious interpretation; it means that substituting an

[2]More precisely, $g(y)$ is not much smaller than $g(x)$, if g is slow-dropping then $g(y)$ cannot be much larger than $g(x)$, either.

approximation to x yields an approximation to $g(x)$. CountSketch returns an approximation to the frequency of every heavy hitter, so, in the first case, we can just as well use the approximation and there is no need to determine the frequency exactly.

The second conclusion allows local variability, but it must be accompanied by some "global" property of g, namely, g does not change a whole lot over the interval $[y, x]$. For an illustration of how predictability works, suppose that H is a heavy hitter and there is some $y \ll f_H$ such that $g(y) \geq 2g(f_H)$. Our algorithm cannot substitute $f_H + y$ for f_H because this could lead to too much error. But, approximating f_H to better accuracy than $\pm y$ using a CountSketch, one would generally need more than $(f_H/y)^2$ counters, which could be very large. Notice that y, if it occurs as a frequency in the stream and suppose it does, is a heavy hitter, and if g is slow-jumping and slow-dropping then y is also a heavy hitter for F_2 by our previous argument. It follows that the error in the frequency estimate derived from the CountSketch is less than y, in particular an estimate of f_H derived from the CountSketch will be more accurate than $\pm y$. The result is that we get an error estimate for the CountSketch that is improved over a naïve analysis. The approximate frequency $\widehat{f}_H$ satisfies $g(\widehat{f}_H) \approx g(f_H)$ and is accurate enough for Braverman and Ostrovsky's algorithm. When g is locally variable but not predictable, we get a one pass lower bound by a reduction from INDEX.

2.4.2 Two Pass Algorithm

We we will now show that any λ-heavy hitter for a slow-dropping, slow-jumping, $\mathbb{S}$-normal function is also F_2 λ'-heavy for λ' modestly smaller than λ. This means that we can use the CountSketch to identify heavy hitters in one pass for the Recursive Sketch. For a heavy hitters algorithm we also need a $(1 \pm \epsilon)$-approximation to each item's contribution to g-SUM, we can easily accomplish this in the second pass by exactly tabulating the frequency of each heavy hitter identified in the first pass. This algorithm works as long as we identify every heavy hitter in the first pass, which CountSketch guarantees, and the space required remains small provided that we do not misclassify too many non-heavy hitters (so that we do not tabulate too many values in the second pass). The procedure is formally defined in Algorithm 1. Note that by Proposition 2.3.7 $\mathbb{P}$-normal functions are also $\mathbb{S}$-normal, hence, the algorithm works for slow-dropping and slow-jumping $\mathbb{P}$-normal functions.

The next two propositions describe equivalent definitions of slow-dropping and slow-jumping that are useful in describing the algorithm.

Proposition 2.4.1. $g \in \mathcal{G}$ is slow-dropping if and only if there exists a sub-polynomial function h such that for all $y \in \mathbb{N}$ and $x < y$ we have $g(x) \leq g(y)h(y)$.

Proof. The "if" direction follows immediately from the definition of the class of sub-polynomial functions.

For the "only if", suppose that $g(x)$ is slow-dropping. Specifically, for any $\alpha > 0$ there exists $N > 0$ such that for all $y \geq N$ and $x < y$ we have $y^{\alpha} g(y) \geq g(x)$. Let N_i be the least integer such that $y^{1/i} g(y) \geq g(x)$, for all $y > N_i$ and $x < y$. The sequence

N_i is nondecreasing. Consider the increasing function $i(y) = \max\{i : N_i \leq y\}$ and set $h(y) = y^{1/i(y)}$. For all $x < y$ we have $h(y)g(y) \geq g(x)$ by construction.

To see that h is sub-polynomial, let $\beta > 0$. First, $h \geq 1$ hence $h(y)y^{\beta} \to \infty$. Second, let $i^* \geq 2/\beta$ then for all $y \geq N_{i^*}$ we have

$$h(y) \leq y^{1/i^*} \leq y^{\beta/2}.$$

Thus $h(y)y^{-\beta} \to 0$, which completes the proof. □

Proposition 2.4.2. $g \in \mathcal{G}$ is slow-jumping if and only if there exists a sub-polynomial function $h(x)$ such that for any $x < y$ we have $g(y) \leq \lfloor y/x \rfloor^2 h(\lfloor y/x \rfloor x)g(x)$.

Proof. Again, the "if" direction follows immediately from the definition of the class of sub-polynomial function. The reverse direction follows from the same argument as Proposition 2.4.1. □

Given g and a sub-polynomial accuracy ϵ, Propositions 2.4.1 and 2.4.2 each imply the existence of a non-decreasing sub-polynomial function. By taking the point-wise maximum, we can assume that these are the same sub-polynomial function $H : \mathbb{N} \to \mathbb{R}$. In particular H satisfies:

- $g(y) \geq g(x)/H(y)$, for all $x < y$, and
- $g(y) \leq (y/x)^2 y^{\alpha} H(y)g(x)$, for all $x < y$.

The space used by our algorithm depends on the sub-polynomial functions governing the

slow-dropping and slow-jumping of g. If these are polylogarithmic then the algorithm is limited to polylogarithmic space.

Lemma 2.4.3. Let g be a function that is slow-jumping and slow-dropping. There exists a sub-polynomial function h such that for any $D \in \mathcal{D}(n, m)$ with frequencies $v_1, v_2, \ldots, v_n$, if $g(v_i) \geq \lambda \sum_j g(v_j)$ then

$$v_i^2 \geq \frac{\lambda}{h(|v_i|)} \sum_{|v_j|<|v_i|} v_j^2.$$

Proof. By Proposition 2.4.2, there exists a nondecreasing sub-polynomial function h such that for all $y \in \mathbb{N}$ and $x < y$ we have $g(y) \leq (y/x)^2 h(y) g(x)$.

For any j that satisfies $|v_j| < |v_i|$, we have $g(v_i) \leq g(v_j)(\frac{v_i}{v_j})^2 h(|v_i|)$. Therefore,

$$g(v_i) \geq \lambda \sum_j g(v_j) \geq \lambda \sum_{|v_j|<|v_i|} \frac{g(v_i)}{h(|v_i|)} \left(\frac{v_j}{v_i}\right)^2.$$

By rearranging, we find

$$v_i^2 \geq \frac{\lambda}{h(|v_i|)} \sum_{|v_j|<|v_i|} v_j^2 \geq \frac{\lambda}{h(|v_i|)} \sum_{|v_j|<|v_i|} v_j^2.$$

□

Algorithm 1 A 2-pass $(g, \lambda, 0, \delta)$-heavy hitters algorithm.

1: **procedure** 2-Pass Heavy Hitters(g, λ, ϵ, δ)
2: **First Pass:**
3: $S \leftarrow$CountSketch($\frac{\lambda}{2H(M)}, \frac{1}{3}, \delta$) discarding the frequency estimates
4: **Second Pass:**
5: Compute v_j for all $i \in S$
6: **return** (j, v_j), for all $j \in S$

Algorithm 1 computes CountSketch[3] with $r = O(\log n)$ and $b = O(\frac{H(M)}{\lambda\epsilon^2})$ over the stream and extracts the $2H(M)/\lambda$ items with the highest estimated frequencies. Denote this list as $(i_j, \widehat{v}_{i_j})_{j=1}^{2H(M)/\lambda}$. The 2-pass algorithm then discards the estimated frequencies $\widehat{v}_{i_j}$ and tabulates the true frequency of each item i_j on the second pass.

Lemma 2.4.4. If g is a function that is slow-dropping and slow-jumping then the CountSketch used by Algorithm 1 finds all of the (g, λ)-heavy hitters.

Proof. From the slow-dropping condition and Lemma 2.4.3, for every $x < y \in \mathbb{N}$ we have

1. $g(y)H(y) \geq g(x)$ and

2. $g(v_i) \geq \lambda \sum_j g(v_j)$ implies
$$v_i^2 \geq \frac{\lambda}{H(M)} \sum_{|v_j|<|v_i|} v_j^2.$$

Let $D \in \mathcal{D}(n, m)$, suppose i satisfies $g(v_i) \geq \lambda \sum_j^n g(v_j)$. Since

$$g(v_i) \geq \lambda \sum_{|v_j|\geq|v_i|} g(v_j) \geq \frac{\lambda}{H(M)} \sum_{|v_j|\geq|v_i|} g(v_i),$$

there are at most $H(M)\lambda^{-1}$ items with frequencies as large or larger than v_i in magnitude.

Thus, a CountSketch for $\lambda/2H(M)$-heavy hitters, as is used by Algorithm 1, serves to identify the λ-heavy heavy hitters for g in one pass. □

Theorem 2.4.5. A function $g \in \mathcal{G}$ is 2-pass tractable and $\mathbb{S}$-normal if it is slow-dropping and slow-jumping.

[3] A CountSketch is a matrix with r and b rows. See[67] for details about these parameters.

Proof. If a function is slow-dropping then it is $\mathbb{S}$-normal. By Theorem 2.3.10, it is sufficient to show that Algorithm 1 is a sub-polynomial memory $(g, \lambda, \epsilon, \delta)$-heavy hitters algorithm for $\lambda = \frac{\epsilon^2}{\log^3 n}$, $\epsilon = \frac{1}{H(Mn)}$, and $\delta \leq \frac{1}{\log n}$.

This is a 2-pass $(g, \lambda, 0, \delta)$-heavy hitters algorithm, and it requires $O(\frac{h(M)}{\epsilon^2} \log^4 n \log nM \log\log n)$ space. The algorithm is correct with probability at least $(1 - \delta)$ because this is true for the CountSketch. Therefore, g is a 2-pass tractable function. □

2.4.3 One Pass Algorithm

The only impediment to reducing the two pass algorithm to a single pass is local variability of the function. The algorithm must simultaneously identify heavy hitters and estimate their frequencies, but local variability could require tighter estimates than seem to be available from a sub-polynomial space CountSketch data structure. If an $\mathbb{S}$-normal function is predictable then we almost immediately overcome this barrier. In particular, the definition means that given any point x and a small error $y = o(x)$ either $|g(x + y) - g(x)| \leq \epsilon g(x)$ or $g(y)$ is reasonably large, and while it is clear how the first case helps us with the approximation algorithm, the second is less clear. The slow-dropping and slow-jumping conditions play a role, as we now explain.

Proposition 2.4.6. A function g is predictable if and only if for every sub-polynomial $\epsilon > 0$ there exists a sub-polynomial function h such that for all $x \in \mathbb{N}$ and $y \in [1, x/h(x))$ satisfying $x + y \notin \delta_{\epsilon(x)}(g, x)$ it holds that $g(y) \geq g(x)/h(x)$.

Proof. The "if" direction follows directly from the definition of sub-polynomial functions.

To prove the "only if" direction, let $\gamma_i = 1/i$. Since g is predictable, there exists N_i such that either $x + y \in \delta_{\epsilon(x)}(g, x)$ or $g(y) \geq x^{-\gamma} g(x)$ for $x \geq N_i$ and $y \in [1, x^{1-\gamma})$. Let $i(z) = \max\{i \in \mathbb{N} \mid N_i \leq z\}$, for $z \geq N_1$, and define $h(z) = z^{1/i(z)}$, for $z \geq N_1$, and $h(z) = \max_{1 \leq x', y' < N_1} g(x')/g(y')$, for $z < N_1$. Then h is a sub-polynomial function. The claim is that h satisfies the conclusion of the lemma. Let $x \in \mathbb{N}$ and $y \in [1, x/h(x))$. If $x < N_1$ then $y \leq x < N_1$ because $h \geq 1$. Furthermore,

$$g(y)h(x) = g(y) \max_{1 \leq x', y' < N_1} \frac{g(x')}{g(y')} \geq g(y) \frac{g(x)}{g(y)} = g(x).$$

If $x \geq N_1$, let $i = i(x) \geq 1$, so that $x \geq N_i$ and $h(x) = x^{1/i}$. Since g is predictable, if $y \in [1, x/h(x))$ and $y + x \notin \delta_{\epsilon(x)}(g, x)$ then $g(y) \geq x^{-1/i} g(x) = g(x)/h(x)$. □

Let $r_\epsilon(x) := \max\{y \mid x + y' \in \delta_\epsilon(x), \text{ for all } |y'| \leq y\}$.

Lemma 2.4.7 (Predictability Booster)**.** If g is slow-dropping, slow-jumping, and predictable, then there is a sub-polynomial function h such that for all $y \in [r_\epsilon(x) + 1, x/h(x))$ we have $g(y) \geq g(x)/h(x)$.

Proof. According to Proposition 2.4.6, there exists a sub-polynomial function h' such that for all $y \in [1, x/h'(x))$ if $x + y \notin \delta_{\epsilon/2}(x)$ then $g(y) \geq g(x)/h'(x)$. Without loss of generality h' governs the slow-jumping of g as well, hence $g(x) \leq (x/x')^2 h'(x) g(x')$ for all $x' \leq x$.

We consider two cases:

1. $|g(x) - g(x + r_\epsilon(x) + 1)| > \epsilon g(x)$ and

2. $|g(x) - g(x - r_\epsilon(x) - 1)| > \epsilon g(x)$.

In the first case, by the definition of r_ϵ and predictability we have $g(r_\epsilon(x)+1) \geq g(x)/h'(x)$. In the second case, let $x' = x - r_\epsilon - 1$. If $g(x') \geq 2g(x)$ then $|g(x') - g(x)| > \epsilon g(x')$, as long as $\epsilon < 1/2$. Otherwise, $|g(x') - g(x)| \geq \epsilon g(x) \geq \frac{\epsilon}{2} g(x')$. Now, from predictability at x' and the definition of h', we find that

$$g(r_\epsilon(x)+1) = g(x - x') \geq \frac{g(x')}{h'(x')} \geq \frac{g(x)}{4h'(x)^2},$$

where the last inequality follows because g is slow-jumping and $x' \geq x/2$ (w.l.o.g., choose $h'(x) > 2$).

Now, since g satisfies the slow dropping condition there exists a nondecreasing sub-polynomial function h'' such that $g(y') \geq g(r')/h''(y')$ for all $y' \in \mathbb{N}$ and $r' \leq y'$. Thus, if $y \in [r_\epsilon(x)+1, x/h(x))$

$$g(y) \geq \frac{g(r_\epsilon(x)+1)}{h''(y)} \geq \frac{g(x)}{4h'(x)^2 h''(x)},$$

so we take h to be the product of $4(h')^2$ and h''. □

Recall the sub-polynomial function H from the last section and let it also satisfy Lemma 2.4.7 with ϵ replaced by $\epsilon/2$. In particular H now satisfies three conditions:

- $g(y) \geq g(x)/H(y)$, for all $x < y$, and
- $g(y) \leq (y/x)^2 H(y) g(x)$, for all $x < y$.
- for all $y \in [r_{\epsilon/2}(x)+1, x/h(x))$ we have $g(y) \geq g(x)/H(x)$.

Let y be the (additive) error in our estimate of frequency x of a (g, λ)-heavy hitter. If

$r_\epsilon(x) > |y|$ then we are fine. If not, consider the point $x + r_\epsilon(x) + 1$ (or $x - r_\epsilon(x) - 1$), which has $|g(x + r_\epsilon(x) + 1) - g(x)| > \epsilon g(x)$ and therefore $g(r_\epsilon(x) + 1) > g(x)/H(M)$. Now, since x is the frequency of a (g, λ)-heavy hitter it happens that $r_\epsilon(x) + 1$, were it a frequency in the stream, would be $(g, \lambda/H(M))$-heavy and thus $\lambda/H(M)^2$-heavy for F_2. Presuming that there are not too many items in the stream with frequency larger than $r_\epsilon(x)$, this implies that CountSketch produces an estimate for x with error smaller than $r_\epsilon(x)$. Now, since g satisfies the slow dropping condition there cannot be too many frequencies larger than $r_\epsilon(x)$ in the stream because

$$g(z) \geq \frac{g(r_\epsilon(x) + 1)}{H(M)} \geq \frac{g(x)}{H(M)^2}, \text{ for all } z > r_\epsilon,$$

which implies that there are at most $\frac{1}{\lambda} H(M)^2$ frequencies in this range.

Algorithm 2 A 1-pass $(g, \lambda, \epsilon, \delta)$-heavy hitters algorithm.

procedure 1-Pass Heavy Hitters(g, λ, ϵ, δ)
  $\widehat{S}, \widehat{V} \leftarrow$CountSketch($\frac{\lambda}{3H(M)}, \frac{\epsilon}{2H(M)}, \delta/2$)
  $\widehat{F}_2 \leftarrow$AMS($\epsilon, \delta/2$)
  $S \leftarrow \{i \in \widehat{S} : |g(\widehat{v}_i) - g(\widehat{v}_i + y)| \leq \epsilon g(\widehat{v}_i + y),$
    for all $-\frac{\epsilon}{2H(M)}\sqrt{\widehat{F}_2} \leq y \leq \frac{\epsilon}{2H(M)}\sqrt{\widehat{F}_2}\}$
  return $(j, \widehat{v}_j)$, for all $j \in S$

2.4.4 One Pass Lower Bounds

The proof of the following theorem is broken up into Lemmas 2.4.9, 2.4.10 and 2.4.11.

Theorem 2.4.8. If $g \in \mathcal{G}$ is a 1-pass tractable $\mathbb{S}$-normal function, then g is slow-jumping, slow-dropping, and predictable.

Proof. Immediate from Lemmas 2.4.9 , 2.4.10, and 2.4.11. □

Lemma 2.4.9. If an $\mathbb{S}$-normal function $g \in \mathcal{G}$ is not slow-dropping, then g is not 1-pass tractable.

Proof. Suppose g is not slow-dropping. Then there exist $0 < \alpha \leq 1$ and integer sequences $y_1, y_2, \ldots \in \mathbb{N}$ and $x_1, x_2, \ldots \in \mathbb{N}$, with y_k increasing, such that $x_k < y_k$ and $g(x_k) \geq y_k^{\alpha} g(y_k)$, for all $k \geq 1$. Furthermore, since g is $\mathbb{S}$-normal we may choose the sequences so that there exists a sub-polynomial function h satisfying

$$|g(x_k + y_k) - g(x_k)| > \frac{1}{h(y_k)} \min\{g(x_k), g(x_k + y_k)\}, \tag{2.4.1}$$

for all k. We claim that one can take $|g(x_k + y_k) - g(x_k)| > g(x_k)/h(y_k)$. If $g(x_k + y_k) < \frac{1}{2} g(x_k)$ then this is true because the constant function 2 is sub-polynomial. On the other hand, if $g(x_k + y_k) \geq \frac{1}{2} g(x_k)$ then replacing h by $2h$ in (2.4.1) does the job. Also, note that $1/h(x)$ is still a sub-polynomial function.

Let $\mathcal{A}$ be a 1-pass (g, ϵ)-SUM algorithm with $\epsilon = (3h(n))^{-1}$. We will show that $\mathcal{A}$ uses $\Omega(n^{\alpha})$ bits on a sequence of g-SUM problems with $n = y_1, y_2, \ldots$, hence g is not tractable. Let $n = y_k$ and $x = x_k$ and consider the following protocol for $\mathsf{INDEX}(n^{\alpha})$ using $\mathcal{A}$. Alice receives a set $A \subseteq [n^{\alpha}]$ and Bob receives an index $b \in [n^{\alpha}]$. Alice and Bob jointly create a notional stream D and run $\mathcal{A}$ on it. Alice contributes n copies of i for each $i \in A$ to the stream and Bob contributes x copies of his index b. If $b \notin A$ there are $|A|$ items in the stream with frequency n and one with frequency x; whereas if $b \in A$ then $|A| - 1$ frequencies are equal to n and one is $n + x$. Alice runs $\mathcal{A}$ on her portion of the stream and sends the

memory to Bob along with the value $|A|$. Bob completes the computation with his portion of the stream. Let $\widehat{G}$ be the value returned to Bob by $\mathcal{A}$. Bob decides that there is an intersection if $\frac{\widehat{G}}{|A|g(n)+g(x)} \notin [1-\epsilon, 1+\epsilon]$.

With probability at least 2/3, $\widehat{G}$ is a $(1 \pm \epsilon)$-approximation to $g(V(D))$; suppose that this is so. If $b \in A$ then $g(V(D)) = (|A|-1)g(n) + g(x+n)$ and otherwise the result is $g(V(D)) = |A|g(n) + g(x)$. Without loss of generality $g(n) < \epsilon g(x)$, thus the difference between these values is

$$\begin{aligned} |g(n) + g(x) - g(x+n)| &> |g(x) - g(x+n)| - \epsilon g(x) \\ &\geq 2\epsilon g(x) \\ &\geq \epsilon(g(x) + n^{\alpha} g(n)) \\ &\geq \epsilon(g(x) + |A|g(n)). \end{aligned}$$

Thus Bob's decision is correct and g-SUM inherits the $\Omega(n^{\alpha})$ lower bound from $\mathsf{INDEX}(n^{\alpha})$.

□

Lemma 2.4.10. If an $\mathbb{S}$-normal function $g \in \mathcal{G}$ is not slow-jumping, then g is not 1-pass tractable.

Proof. Since g is not slow-jumping, there exist $\alpha > 0$, a strictly increasing sequence $y_1, y_2, \ldots \in \mathbb{N}$, and a sequence $x_1, x_2, \ldots \in \mathbb{N}$ such that $x_k \leq y_k$ and $g(y_k) > s_k^{2+\alpha} x_k^{\alpha} g(x_k)$, where $s_k = \lfloor y_k / x_k \rfloor$. Without loss of generality y_k is strictly increasing. According to Lemma 2.4.9, we can assume that g is slow-dropping, since otherwise it is not 1-pass

tractable. Hence, there exists $N \in \mathbb{N}$ such that for all $x \geq N$ and $r \leq x$ we have $\frac{1}{2}x^{\alpha}g(x) \geq g(r)$. If $x_k \leq N$ for all but finitely many k, we may assume this holds for all k. Otherwise, by taking a subsequence, we can assume $x_k > N$ holds for all k.

Let $r_k = y_k - s_k x_k$ be the sequence remainders. Before proceeding with the reduction we will establish the bound $2g(r_k) \leq g(y_k)$. If $x_k \leq N$, for all k, then s_k is unbounded while $x_k g(x_k)$ is bounded away from zero. This means that $g(y_k)$ is unbounded while $g(r_k)$ is bounded and we can assume y_1 is large enough that $g(y_k) \geq 2g(r_k)$ holds for all k. Otherwise $x_k > N$, for every k, hence $2g(r_k) \leq x_k^{\alpha} g(x_k) \leq g(y_k)$, for each k, from the definition of N.

Let $\mathcal{A}$ be a (g, ϵ)-SUM algorithm with accuracy $\epsilon = 1/12$. Consider the $\mathsf{DISJ}{+}\mathsf{IND}(n, t)$ problem, where $t = s_k$ and $n = s_k^{2+\alpha} x_k^{\alpha}$. Denote $x := x_k$, $y := y_k$, $s := s_k$ and $r := r_k$. The $t + 1$ players receive sets $A_1, A_2, \ldots, A_t \subseteq [n]$ and an index $b \in [n]$. As before, the players jointly run $\mathcal{A}$ on a notional stream and share $|A_i|$. Each player i places x copies of each $j \in A_i$ into the stream except for the final player who places r copies of his index b. Let $n' = \sum_i |A_i|$ be the total size of the t players' sets. On an intersecting instance the result of g-SUM is $a_1 := (n' - t)g(x) + g(y)$, and on a disjoint instance the result is $a_2 := n'g(x) + g(r)$.

The difference is

$$\begin{aligned} a_1 - a_2 &\geq g(y) - g(r) - tg(x) > \frac{1}{2}g(y) - tg(x) \\ &\geq \frac{1}{6}(g(y) + (2s^{2+\alpha}x^{\alpha})g(x)) - sg(x) \\ &= \frac{1}{6}(g(y) + (s^{2+\alpha}x^{\alpha} + s^{2+\alpha}x^{\alpha} - 6s)g(x)) \\ &\geq 2\epsilon(g(y) + s^{2+\alpha}x^{\alpha}g(x)) \\ &= 2\epsilon(n'g(x) + g(y)), \end{aligned} \tag{2.4.2}$$

where the last inequality holds for all sufficiently large n. Thus, the index player can correctly distinguish which case has occurred, and the algorithm requires $\Omega(n/t^2) = \Omega(y^{\alpha})$ bits. □

Lemma 2.4.11. If an $\mathbb{S}$-normal function $g \in \mathcal{G}$ is not predictable, then g is not 1-pass tractable.

Proof. Since g is not predictable, there exists a sub-polynomial function ϵ and constant $\gamma > 0$ such that some infinite sequence x_k, y_k satisfies the following:

- $x_k \to \infty$ and $y_k \in [1, x_k^{1-\gamma})$,
- $x_k + y_k \notin \delta_{\epsilon(x_k)}(g, x_k)$, and
- $x_k^{\gamma} g(y_k) < g(x_k)$.

The proof is by a reduction from $\mathsf{INDEX}(n)$ with $n = \epsilon(x_k)x_k^{\gamma}/4$. Suppose $\mathcal{A}$ is an algorithm for $(g, \epsilon/4)$-SUM. Alice receives a set $A \subseteq [n]$ and Bob receives an index $b \in [n]$. Alice adds

y_k copies of i to the stream, for each $i \in A$, runs $\mathcal{A}$ on her portion of the stream, and sends the contents of the memory to Bob. Bob adds x_k copies of b to the stream and completes the computation.

The stream has $|A|$ frequencies equal to y_k and one equal to x_k, if there is no intersection, or $|A| - 1$ equal to y_k and one equal to $x_k + y_k$, if there is an intersection. Recall that $x_k + y_k \notin \delta_\epsilon(g, x_k)$, hence $|g(x_k) - g(x_k + y_k)| > \epsilon g(x_k)$, and by construction

$$|A|g(y_k) \leq \frac{\epsilon(x_k)}{4} x_k^\gamma g(y_k) \leq \frac{\epsilon(x_k)}{4} g(x_k).$$

Therefore, Bob can correctly distinguish whether $b \in A$ when $\mathcal{A}$ yields a $(1 \pm \epsilon/4)$-approximation. Thus $(g, \epsilon/4)$-SUM requires $\Omega(n)$ bits. □

2.4.5 Two Pass Lower Bounds

Theorem 2.4.12. If a $\mathbb{P}$-normal function $g \in \mathcal{G}$ is tractable then g is slow-dropping and slow-jumping.

Proof. Immediate from Lemmas 2.4.13 and 2.4.14. □

Lemma 2.4.13. If a $\mathbb{P}$-normal function $g \in \mathcal{G}$ is not slow-dropping, then g is not $O(1)$-pass tractable.

Proof. Suppose $\mathcal{A}$ is a $p = O(1)$ pass algorithm that solves (g, ϵ)-SUM and g does not satisfy slow-dropping condition. Then there exists $\alpha > 0$ such that for any $N > 0$, there exists $y > N$ and $x < y$ satisfying $g(y) < g(x)/y^\alpha$. Separately, since g is $\mathbb{P}$-normal, there exists

$\beta > 0$ such that for any $N > 0$ if g has an α-period $y > N$ then there is an α-period $y > N$ and $x < y$ such that $g(y) \leq g(x)/y^{\alpha}$ and $|g(x+y) - g(x)| > y^{\beta} \min(g(x), g(x+y))$.

Let $\gamma = \min(\alpha, \beta)$. Consider the following protocol for DISJ$(n, 2)$, where $n = y^{\gamma/2}$.

First, consider $g(x+y) \leq g(x)$. Since $|g(x+y) - g(x)| > y^{\gamma} g(x+y)$ and $y^{\gamma} g(x+y) \geq g(x+y)$, then $g(x) \geq y^{\gamma} g(x+y)$. The players jointly create a stream where Player 1 inserts x copies of each element of her set S_1 into the stream, and Player 2 inserts y copies of every element a not in her set S_2. First Player 1 creates her portion of the stream, runs the first pass of $\mathcal{A}$ on it, and sends the memory to Player 2. She completes the first pass with her portion of the stream, and returns the memory to Player 1. The players continue in this way for a total of p passes over the stream.

Let V denote the frequency vector of the resulting stream. If there is an intersection, let $S_1 \cap S_2 = \{a\}$. Then $S_1 \cap \bar{S}_2 = S_1 \backslash \{a\}$, and $g(V)$ is $r_1 = (|S_1| - 1)g(x+y) + (n - |S_2| - |S_1|)g(y) + g(x) + g(y)$. If there is no intersection, the value of $g(V)$ is $r_2 = (|S_1| - 1)g(x+y) + (n - |S_2| - |S_1|)g(y) + g(x+y)$. Notice that

$$\frac{|r_2 - r_1|}{r_1} \geq \frac{|g(x+y) - g(x)|}{g(x)} \geq \frac{1}{2},$$

for sufficiently large n. For any $\epsilon < 1/2$, $\mathcal{A}$ is able to distinguish the 2 cases, which gives a lower bound on the memory bits used by $\mathcal{A}$ is $\Omega(n/p)$.

The case $g(x+y) > g(x)$ is the same except Player 2 inserts y copies of each element that *is* in her set S_2. Thus, $\mathcal{A}$ uses $\Omega(n/p)$ bits of memory, hence g is not $O(1)$-pass

tractable. □

Lemma 2.4.14. If a $\mathbb{P}$-normal function $g \in \mathcal{G}$ is not slow-jumping, then g is not $O(1)$-pass tractable.

Proof. Suppose $\mathcal{A}$ is a $p = O(1)$ pass algorithm for (g, ϵ)-SUM. If g is not slow-jumping, then there exists $\alpha > 0$ such that for any $N > 0$, there exists $x < y \in \mathbb{N}$ and $y \geq N$ but $g(y) > \lfloor y/x \rfloor^{2+\alpha} x^\alpha g(x)$. Notice that for $y > x$ we have $\lfloor y/x \rfloor \geq y/2x$, so by adjusting α we can assume $g(y) > (y/x)^2 y^\alpha g(x)$.

We can further assume g is slow dropping, since otherwise g is not $O(1)$-pass tractable by Lemma 2.4.13. Thus, there exists a nondecreasing sub-polynomial function $h(x) > 1$ such that $g(x) \leq h(y)g(y)$.

Consider an instance $A_1, A_2, \ldots, A_t \subseteq [n]$ of $\mathsf{DISJ}(n, t)$, where $t = \lceil y/x \rceil$ and $n = (\frac{y}{x})^2 \frac{y^\alpha}{2h(y)}$. Each of the first $t-1$ players, inserts x copies of her elements into the stream and the tth player inserts $y - (t-1)x < x$ copies of her elements into the stream. The players pass the memory and repeat to run the p passes of $\mathcal{A}$ as before. Notice that $g(y - (t-1)x) \leq h(x)g(x)$ by the slow dropping condition.

Let $n' = \sum_i |A_i|$, and let V be the frequency vector of the stream created. If there is no intersection, the value of $g(V)$ is

$$\begin{aligned}(n' - |A_t|)g(x) + |A_t|g(y - (t-1)x) &\leq n'h(x)g(x) \\ &\leq nh(x)g(x) \leq \frac{1}{2}g(y).\end{aligned}$$

On the other hand, if there is an intersection then y is the frequency of some item in the stream, and therefore $g(V) \geq g(y)$. Thus $\mathcal{A}$ distinguishes between the cases. The algorithm uses $\Omega(n/pt^2) = \Omega(y^{\alpha/2}/p)$ bits of memory, which proves that g is not p-pass tractable. $\square$

2.4.6 Zero-One Law Proofs

Theorem 2.2.1 1-pass Zero-One Law. A function $g \in \mathcal{G}$ is 1-pass tractable and $\mathbb{S}$-normal if and only if it is slow-jumping, slow-dropping, and predictable.

Proof. The lower bound is proved as Theorem 2.4.12, hence the algorithm remains. With the Recursive Sketch of Theorem 2.3.10, it is enough to show that Algorithm 2 is a $(g, \lambda, \epsilon, \delta)$-heavy hitters algorithm for $\lambda = \epsilon^2/\log^3 n$ and $\delta = 1/\log n$.

By Lemma 2.4.4, the Count Sketch used by Algorithm 2 returns a list of pairs $(i_j, \widehat{v}_{i_j})$ containing all of the $\lambda/3H(M)$-heavy elements for F_2. By definition, any item i_j that survives the pruning stage has $|g(\widehat{v}_{i_j}) - g(v_j)| \leq \epsilon g(v_j)$. Hence, it only remains to show that every (g, λ)-heavy hitter survives the pruning stage.

Suppose i' is the index of the (g, λ)-heavy hitter that minimizes $r_{\epsilon/2}(v_{i'})$. Now suppose that we insert a new item i'' in the stream with frequency $v_{i''} = r_{\epsilon/2}(v_{i'}) + 1$. Then

$$\frac{g(v_{i'})}{H(M)} \leq g(v_{i''}) \leq H(M)g(v_{i'}),$$

where the first inequality follows from Lemma 2.4.7 and the second because g is slow-dropping. Thus, this item is a $(g, \frac{\lambda}{3H(M)})$-heavy hitter and the constant term on the pa-

rameter b for the Count Sketch can be chosen to guarantee additive error no more than $v_{i''}/3 \leq r_{\epsilon/2}(v_{i'})/3$. The same guarantee holds for the stream without i'', thus for every (g, λ)-heavy hitter i_j we have

$$|\widehat{v}_{i_j} - v_{i_j}| \leq \frac{1}{3} r_{\epsilon/2}(v_{i'}) \leq \frac{1}{3} r_{\epsilon/2}(v_{i_j}),$$

where the last inequality holds by the choice of i'. Furthermore, for every $-r_{\epsilon/2}(v_{i'})/3 \leq y \leq r_{\epsilon/2}(v_{i'})/3$ we have $|(\widehat{v}_{i_j} + y) - v_{i_j}| \leq r_{\epsilon/2}(v_{i_j})$ and thus

$$\begin{aligned} |g(\widehat{v}_{i_j}) - g(\widehat{v}_{i_j} + y)| \leq & |g(\widehat{v}_{i_j}) - g(v_{i_j})| \\ & + |g(v_{i_j}) - g(\widehat{v}_{i_j} + y)| \\ \leq & \epsilon g(\widehat{v}_{i_j} + y). \end{aligned}$$

With probability at least $1 - 2\delta/2 = 1 - \delta$ both the Count Sketch and the AMS approximation of F_2 meet their obligations, in this case the output is correct. By the guarantee of CountSketch, we can safely assume that $r_{\epsilon/2}(v'_i) \geq \epsilon/2H(M)\sqrt{\widehat{F}_2}$, heavy hitters will survive the pruning stage. □

Theorem 2.2.2 2-pass Zero-One Law. A function $g \in \mathcal{G}$ is 2-pass tractable and $\mathbb{P}$-normal if and only if it is slow-dropping and slow-jumping. Furthermore, every slow-dropping and slow-jumping $\mathbb{S}$-normal function is also 2-pass tractable.

Proof. The lower bound is proved as Theorem 2.4.12. The upper bound is governed by Proposition 2.3.7 and Theorem 2.4.5. □

Here are a few examples. The functions $x^2 \lg(1+x)$, $(2 + \sin \log(1 + x))\, x^2$, and $e^{\log^{1/2}(1+x)}$ are all 1-pass tractable because they are all slow-dropping, slow-jumping, and predictable. On the other hand, $1/x$ is not slow-dropping, x^3 is not slow-jumping, and $(2 + \sin \sqrt{x})\, x^2$ is not predictable, so none of these functions is 1-pass tractable. The last of the three is, however, slow-jumping and slow-dropping, and hence it is 2-pass tractable.

2.5 Nearly Periodic Functions

Nearly periodic functions are highly constrained to *almost* repeat themselves like a periodic function. These functions admit large changes in value that would imply a large lower bound on their space complexities, if they did not satisfy the many constraints.

2.5.1 Example Nearly Periodic Function

Constructing a tractable nearly periodic function turns out to be a non-trivial exercise. This section provides such a construction.

Definition 2.5.1. Let $x = \sum_{j=0}^{\infty} a_j 2^j \in \mathbb{N}$, for $a_j \in \{0, 1\}$, be the binary expansion of the integer x. Define $i_x := \min\{j : a_j = 1\}$, the location of the first non-zero bit of x. The function is $g_{np}(x) = 2^{-i_x}$, for $x > 0$, and $g_{np}(0) = 0$.

Proposition 2.5.2. g_{np} is $\mathbb{S}$-nearly periodic.

Proof. Since there is an infinite sequence $\{1, 2, 4, \ldots, 2^n \ldots\}$ such that $g_{np}(x) = 1/x$, the first condition of near-periodicity is satisfied. It is remains to show that the second condition

is also satisfied.

Let $\gamma > 0$ be a constant. Consider any integer $x > 0$. By construction $g_{np}(x) = 2^{-i_x}$. Let $y > x$ be another integer. Similarly, we have $g_{np}(y) = 2^{-i_y}$. If $g_{np}(x)/g_{np}(y) = 2^{i_y - i_x} \geq y^\gamma$, then $i_y - i_x \geq \gamma \lg y$. Choose $N = \lceil 2^{1/\gamma} \rceil$. For any $y > N$, we have $i_y - i_x > 1$ and therefore $i_{x+y} = i_x$. Thus, $g_{np}(x + y) = g_{np}(x)$. □

Proposition 2.5.3. g_{np} is 1-pass tractable.

Proof. We demonstrate that one can find (g_{np}, λ)- heavy hitters in $\text{poly}(\lambda^{-1} \log n \log M)$ space. It is sufficient to demonstrate an algorithm that will find a single (g_{np}, λ)-heavy hitter, since we can reduce the heavy hitters problem to this case by hashing the stream $O(\lambda^{-2})$ ways and running this algorithm on each substream.

Suppose that j^* is the single (g_{np}, λ)-heavy hitter with frequency x. Let $v_1, v_2, \ldots, v_n \geq 0$ be the frequencies in the stream, and let $U = \{j : i_{v_j} \leq i_x\}$. By definition $g_{np}(v_j) \geq g_{np}(x)$ for $j \in U$, hence $|U| \leq 1 + \lambda^{-1} \leq 2\lambda^{-1}$.

Let $C = O(\lambda^{-2})$. Apply a uniform hash function $h : [n] \to [C]$ to separate the stream into C substreams $S_1, S_2, \ldots, S_C$. With constant probability, no two elements of U are in the same substream, so suppose that this happens.

On each substream S_k with frequencies $v_1^{(k)}, \ldots, v_n^{(k)}$ we run the following algorithm $D = O(\log n)$ times independently in parallel: (i) Sample pairwise independent Bernoulli(1/2) random variables $X_1, \ldots, X_n$. (ii) Compute $m = \sum_j X_j v_j^{(k)}$ and i_m. (iii) Output 2^{-i_m}.

Consider the set of D values output by this algorithm. If there is a single item j^* with

minimum $i^* := i_{v_{j^*}^{(k)}}$, then a Chernoff bound implies that very nearly $D/2$ values are equal to 2^{-i^*}, and this is the maximum value among all of the pairs. In this case, the label j^* can be found in post-processing by binary search. Let $X_{j,\ell}$ denote the value of the jth Bernoulli variable on the ℓth trial. Specifically, one finds the set $M \subseteq [D]$ of trials for which 2^{-i_m} is equal to the maximum among the D values, and with high probability, only j^* will satisfy $X_{j,\ell} = 1$ for all $\ell \in M$ and $X_{j,\ell} = 0$ for all $\ell \in [D] \setminus M$. We can detect the case where there is more than one item with minimum $i_{v_j^{(k)}}$ because either the number of maximizing values will be too large or the binary search will fail to yield a unique element.

The single-heavy-hitter algorithm now outputs the pair $(j^*, 2^{-i^*})$ if the number of maximizing values is correct and the binary search yields a unique element or nothing if either of those conditions fails. With the extra hashing step mentioned at the beginning this yields a 1-pass $O(\lambda^{-4} \log n \log M)$-space (g_{np}, λ)-heavy hitters algorithm, thus g_{np} is 1-pass tractable. □

2.5.2 More Lower Bounds

The set of nearly-periodic function defeat the standard reduction from the DISJ and INDEX problems. Theorem 2.5.7 shows how one can use the ShortLinearCombination problem, of Definition 2.3.11, to provide space lower bounds on g-SUM algorithms for some nearly periodic functions g.

Definition 2.5.4. For any $N > 0$ and non-increasing sub-polynomial function $h(x)$, define the (N, h)-dropping set of f to be $\mathcal{D}_{N,h}(f) = \{x \mid 1 \leq x \leq N, f(x) \leq h(N)/N\}$.

Proposition 2.5.5. Let f be a nearly-periodic function, then for all $n_0 > 0$, there exists

$N > n_0$ and a sub-polynomial function h such that $|D_{N,h}| > 0$.

Proof. Choose $N = n_0 + 1, h(x) = f(1)N$, i.e. $h(x)$ is a constant function. Then 1 in $D_{N,h}$ because $1 \leq N$ and $f(1) = h(N)/N$. □

Definition 2.5.6. An *α-indistinguishable* frequency set of $[n]$ is a tuple (s, d), where $s \subseteq [n]$ is a subset and $d \in [n], d \notin s$ is a integer such that (s, d)-DIST problem requires $\Omega(n^\alpha)$ space.

Theorem 2.5.7. Let function $f : \mathbb{R} \to \mathbb{R}^+$, symmetric, nearly-periodic and $f(0) = 0$. If there exists a non-increasing sub-polynomial function $h(x)$ and constants $\alpha > 0, 0 < \delta < 1$, such that for any $n_0 > 0$, there exists $N > n_0$ and a integer set $|S| > 0$ satisfying the follows,

1. $0 < |\mathcal{D}_{N,h}(f)| \leq N^{1-\delta}$;
2. $S \subseteq |\mathcal{D}_{N,h}(f)|$ and $d \in [N]$ such that (S, d) is a α-indistinguishable frequency set of N;
3. $f(d) \geq h(d)$,

then for any $n_0 > 0$, there exists a stream of domain $N > n_0$, any one-pass randomized streaming algorithm $\mathcal{A}$ that outputs a $1 \pm \epsilon(N)$ approximation of f-SUM with probability at least $2/3$, requires space $\Omega(n_0^\alpha)$ bits, where $\epsilon(x) < 1$ is a sub-polynomial function.

Proof. Suppose we have an algorithm $\mathcal{A}$ that gives a $1 \pm \epsilon(N)$ approximation to f-SUM with probability at least $2/3$. We can now use $\mathcal{A}$ to construct a protocol that solves (S, d)-DIST problem on domain N. The algorithm is straightforward: run $\mathcal{A}$ on the input stream. If the result $\mathcal{A}$ outputs is $\leq h(d)/N^\delta$, then there is no frequency of absolute value d in the stream.

This is so since the g-SUM is at most $\sum_{a \in S} f(a) \leq N^{1-\delta}/N$. Otherwise, if the output is at least $h(d)(1-\epsilon(N))$. By the guarantee of $\mathcal{A}$, it can distinguish the 2 cases. The space needed of $\mathcal{A}$ inherits the lower bound of (S,d)-DIST problem, thus $\Omega(N^{\alpha})$ bits. □

2.6 Appendix for Chapter 2

2.6.1 The Case of $g(0) \neq 0$

When $g(0) = 0$, we have to re-address the lower bound of using INDEX since the elements not contributing to the frequency vector still contribute to the g-SUM. Therefore, we re-define the nearly periodic functions in the next section. We further show that functions crossing the axis are not 1-pass tractable and functions with zero points are not tractable unless they are periodic in Section 2.6.3. We prove the same 1-pass zero one law in the remaining 2 sections. The proof turns out to be very similar to the $g(0) = 0$ cases with small number of additional tweaks. Note that we only provide the turnstile model lower bounds.

2.6.2 Redefinition of Nearly Periodic

For the $g(0) \neq 0$ case, we have the following definition of nearly-periodic.

Definition 2.6.1. Given a set of functions $\mathcal{S}$, call $g(x)$ *$\mathcal{S}$-nearly periodic*, if the following two conditions are satisfied.

1. There exists $\alpha > 0$ such that for any constant $N > 0$ there exists $x, y \in \mathbb{N}$, $x < y$ and

$y \geq N$ such that $g(y) \leq g(x)/y^{\alpha}$. Call such a y *α-period of g*;

2. For any $\alpha > 0$ and any error function $h \in \mathcal{S}$ there exists $N_1 > 0$ such that for all α-periods $y \geq N_1$ all $x < y$ such that $g(y)y^{\alpha} \leq g(x)$ we have $|g(x) - g(x-2y)| \leq \min\{g(x), g(x-2y)\}h(y)$.

A function g is *$\mathcal{S}$-normal* if it is not $\mathcal{S}$-nearly periodic.

Let all functions be $\mathcal{G}_0^* = \{g : \mathbb{N} \to \mathbb{R}^+, g(x) = g(-x), \text{and } g(0) = 1\}$.

2.6.3 Crossing the Axis

If $g \in \mathcal{G}_0^*$ and there exist $x, y \in \mathbb{N}$ such that $g(x) > 0 > g(y)$, we have the following lemma.

Lemma 2.6.2. Let $g \in \mathcal{G}_0^*$, if $g(1) < 0$, then any algorithm solves g-SUM requires $\Omega(n)$ space.

Proof. Consider the following INDEX(n) reduction. Alice and Bob jointly create a stream as following. Let A be Alice's set. $n_1 = |A|$. Let $n_2 = \lfloor \frac{n-n_1}{-g(1)} \rfloor$. Let $C = (n_1+n_2)g(1)+(n-n_1)$. We have $0 \leq C < g(1)$. Alice choose a domain $[n+n_2]$ for the algorithm and output 1 copy of each of her element. Bob receives the memory content of the algorithm from Alice as well as n, n_1, n_2. Bob outputs -1 copy of his index and also adds to the stream one copy of each of the following elements: $n+1, n+2, \ldots, n+n_2$.

If there is an intersection, the result of g-SUM is

$$(n - n_1) + (n_1 + n_2 - 1)g(1) + g(0) = C + g(0) - g(1);$$

If there is no intersection, the result is

$$(n - n_1 - 1) + (n_1 + n_2)g(1) + g(1) = C + g(1) - g(0).$$

Then a constant approximation algorithm can distinguish the two cases. The algorithm inherits an $\Omega(n)$ bound from INDEX. □

Lemma 2.6.3. Let $g \in \mathcal{G}_0^*$ and $g(1) > 0$, if there exists $y \in \mathbb{N}$ such that $g(y) < 0 < g(1)$, then there exits $z \in \mathbb{N}$ such that $g(1 + z) \neq g(1 - z)$.

Proof. If for every z, $g(1 + z) = g(1 - z)$, we have $g(2) = g(0) = 1$ and for any w, $g(w+2) = g(w)$. Therefore 2 is a period of the function. Now, consider the following cases, 1) if y even, then $g(y) = g(2\lfloor y/2 \rfloor)) = g(0)$ (2 is a period), which is a contradiction that $g(y) < 0$; 2) if y odd, then $g(y) = g(2\lfloor y/2 \rfloor + 1) = g(1) > 0$, contradicting $g(y) < 0$. □

Proposition 2.6.4. Let $g \in \mathcal{G}_0^*$ and $g(1) > 0$, if there exists $y \in \mathbb{N}$ such that $g(y) < 0$, then any algorithm solves g-SUM requires $\Omega(n)$ space.

Proof. Consider the following reduction from INDEX(n). Let $n' = \lfloor -(n-1)g(1)/g(y) \rfloor$ and $z \in N$ be the integer given by Lemma 2.6.3. Let $C = (n-1)g(1) + n'g(y)$. We have $0 < C < -g(y)$. Alice and Bob jointly create a data stream in domain $n+n'$: Alice outputs 1 copy of every element in her set, and -1 copy of every element not in her set. Bob outputs

y copy of each of the elements $i+1, i+2, \ldots, i+n'$ and z copy of his index. If there is no intersection, the result of g-SUM is

$$(n-1)g(x) + n'g(y) + g(z-1) = C + g(z-1).$$

If there is an intersection, the result of g-SUM is

$$(n-1)g(x) + n'g(y) + g(z+1) = C + g(z+1)$$

Therefore any constant approximation algorithm can distinguish the two cases, implying an $\Omega(n)$ bound for the memory of the algorithm. □

Proposition 2.6.5. Let $g \in \mathcal{G}_0^*$ if $g(x) = 0$ and $g(2x) \neq g(0)$ then g is not 1-pass tractable.

Proof. Consider the INDEX(n) reduction. Alice and Bob jointly create a stream as following, Alice output x for every element in her set, $-x$ for every element not in her set. Bob output x for his index. If there is an intersection, the g-SUM result is $g(2x)$. If there is no intersection, the result is $g(0)$. □

Proposition 2.6.6. If function $g \in \mathcal{G}_0^*$ with $g(x) = 0$ for some $x \in \mathbb{N}$ is tractable, then g is periodic.

Proof. By reduction from INDEX(n), Alice output x for each of her elements and $-x$ for each of her non-elements. Bob output $k + x$ for his index. If there is an intersection, the g-SUM result is $g(k+2x)$. If there is no intersection, g-SUM is $g(k)$. If $g(k+2x) \neq g(k)$

for any k, a constant approximation algorithm can distinguish the 2 cases. Therefore, $2x$ is a period of the function. □

Therefore, we only consider the following set of functions,

$$\mathcal{G}_0 = \{g : \mathbb{N} \to \mathbb{R}^+, g(x) = g(-x) > 0, \text{and } g(0) = 1\}$$

2.6.4 Lower Bounds

Theorem 2.6.7. If an $\mathbb{S}$-normal function $g \in \mathcal{G}_0$ does not satisfy slow dropping, then g is not 1-pass tractable.

Proof. The reduction to INDEX(n^α) is similar to the case of $g(0) = 0$. Now Alice outputs n copies of the elements in A and also outputs $-n$ copies of i if $i \notin A$. Bob output $x-n$ copies of his index. If there is an intersection the g-SUM result is $g(V(D)) = (n^\alpha - 1)g(n) + g(x)$ and otherwise is $g(V(D)) = (n^\alpha - 1)g(n) + g(x - 2n)$. Since g is not nearly periodic, the algorithm is able to distinguish the two cases. □

Theorem 2.6.8. If an $\mathbb{S}$-normal function $g \in \mathcal{G}_0$ is not slow-jumping, then g is not 1-pass tractable.

Proof. The reduction to DISJ+IND is the same to the case of $g(0) = 0$.

Now by slow dropping, there exists a sub-polynomial function $h(x)$ such that $g(0) \leq g(x)h(x)$. The g-SUM result differs from $g(0) = 0$ case by $(n - n')g(0)$ (or $(n - n' + t)g(0)$), which is comparable to $(n - n')g(x)$. Therefore, we can apply the same analysis. □

Theorem 2.6.9. If an $\mathbb{S}$-normal function $g \in \mathcal{G}_0$ is not predictable, then g is not 1-pass tractable.

Proof. Still use the same reduction to INDEX with the $g(0) = 0$ case. Now the g-SUM results differs by $(n - |A|)g(0)$. By slow dropping, there exists a sub-polynomial function $h(x)$ such that $g(0) \leq h(y)g(y)$. Therefore, all the analyses remain the same. □

2.6.5 Upper Bound

The algorithm should still work by the following 2 lemmas.

Lemma 2.6.10. Let $g \in \mathcal{G}_0$ be an $\mathbb{S}$-normal function that is slow-jumping and slow-dropping. There exists a sub-polynomial function h such that for any $D \in \mathcal{D}(n, m)$ with frequencies $v_1, v_2, \ldots, v_n$, if $g(v_i) \geq \lambda \sum_j g(v_j)$ then $v_i^2 \geq \frac{\lambda}{h(|v_i|)} \sum_{|v_j|<|v_i|} v_j^2$.

Proof. The proof is the same as in the $g(0) = 0$ case with additional care of $g(0) = 1$. □

Lemma 2.6.11. If $g \in \mathcal{G}_0$ is slow-dropping, slow-jumping, and predictable, then there is a sub-polynomial function h such that for all $y \in [r_\epsilon(x)+1, x/h(x))$ we have $g(y) \geq g(x)/h(x)$.

Proof. The proof is the same as in the $g(0) = 0$ case with additional care of $g(0) = 1$. □

2.6.6 Lower bound for DISJ+IND

Theorem 2.6.12. The randomized one-way communication complexity of $\mathsf{DISJ{+}IND}(n, t)$ is $\Omega(n/t \log n)$.

Proof. We give a reduction from $\mathsf{DISJ}(n, t+1)$. Let $\mathcal{P}$ be any randomized protocol for $\mathsf{DISJ}{+}\mathsf{IND}(n,t)$. Run in parallel $\ell = \lceil 96 \log n \rceil$ independent copies of $\mathcal{P}$ through the first t players. This produces ℓ transcripts. Player $t+1$ now takes the each of the transcripts and computes the final value of each once for every element he holds, as if it was the only element he held. No communication is need for this part because $\mathcal{P}$ is a one-way protocol and $t+1$ is the final player.

Player $t+1$ then takes the majority vote among the independent copies of $\mathcal{P}$ for each element. If any vote signals an intersection then he reports intersection; otherwise he reports disjoint.

If every one of the $|A_{t+1}| \leq n$ majorities is correct then the final player's report is correct. Let X_i, for $i \in A_{t+1}$, be the number among the ℓ copies of $\mathcal{P}$ with the correct outcome when the final player completes protocol using $i \in A_{t+1}$. Then X_i is Binomially distributed from ℓ trials with success probability at least $2/3$. Using a Chernoff Bound we find

$$\begin{aligned} P(X_i \leq \ell/2) = P\left(X_i \leq (1-\frac{1}{4})\frac{2}{3}\ell\right) &\leq \exp\left\{\frac{-1}{32}\mu\right\} \\ &\leq \exp\left\{\frac{-1}{32}\cdot\frac{2}{3}\ell\right\} \leq \frac{1}{n^2}. \end{aligned}$$

Thus, with probability at least $1 - \frac{1}{n}$ every majority vote is correct, hence our $\mathsf{DISJ}(n,t)$ protocol is correct for $n \geq 3$.

Let $T_1, T_2, \ldots, T_\ell$ be the transcripts. The total cost of this DISJ protocol is $\sum_{i=1}^{\ell} |T_i|$. Since $\mathsf{DISJ}(n, t+1)$ requires $\Omega(n/t)$ bits of communication, at least one of the protocols has

length $\Omega(n/t\ell) = \Omega(n/t \log n)$, hence $|\mathcal{P}| = \Omega(n/t \log n)$ bits of communication. □

2.6.7 Lower bound for ShortLinearCombination

The 3-frequency Distinguishing Problem

We first define the ShortLinearCombination problem for three frequencies, which we call (a, b, c)-DIST for short.

Definition 2.6.13. A stream S with frequency vector v is given to a one-pass streaming algorithm. v is promised to be from $V_0 = \{-a, a, -b, b, 0\}^n$ or $V_1 = \{\text{EMB}(v, i, e) \mid v \in V_0, i \in [n], e \in \{-c, c\}\}$, where $a, b, c > 0$ and

$$\text{EMB}(v, i, e)_j = \begin{cases} v_i & i \neq j \\ e & i = j \end{cases}$$

(in other words, V_1 is given by replacing a coordinate of a vector v from V_0 with $-c$ or c). The (a, b, c)-DIST problem is for the algorithm to distinguish whether $v \in V_0$ or $v \in V_1$.

To study this problem, we first consider the special case where $c = \gcd(a, b) = 1$.

Proposition 2.6.14. If $\gcd(a, b) = 1$, then any randomized algorithm solving $(a, b, 1)$-DIST with probability at least $2/3$ requires $\Omega(n/\max(a, b)^2)$ bits of memory.

Proof. We use the same notation as in,[17] and in particular refer to that work for background on information complexity. Without loss of generality, we assume $a > b$. Consider a communication problem in which Alice is given a vector v_1 and Bob is given a vector v_2,

for which $v = v_1 - v_2 \in V_0 \cup V_1$. The players are asked to solve the $(a, b, 1)$-DIST on v. This problem is OR-decomposable with primitive $\mathrm{DIST}(x, y)$, where $\mathrm{DIST}(x, y) = 1$ if and only if $|x - y| = 1$. Consider a randomized protocol Π which succeeds with probability at least $2/3$ in solving this problem, and suppose Π has the minimum communication cost of all such protocols. We now call $(a, b, 1)$-DIST f for short. The following is well known (see, e.g.,[17]),

$$R_{2/3}(f) \geq IC_{\mu,2/3}(f),$$

where $R_{2/3}(f)$ is the communication complexity of the best randomized protocol (with error probability at most $2/3$) on the worst case input, μ is any distribution over the input space, and IC_μ is the information complexity of the best protocol over the input distribution μ. We now construct an input distribution μ as follows. Let D be chosen uniformly at random over {Alice, Bob}, E is chosen uniformly from $\{a, a+1, a+2, \ldots, m-a\}$, where $m \gg a$ is a sufficiently large integer for which the coordinates of the vector jointly created by Alice and Bob will never exceed m. Denote by ξ the joint distribution of (D, E). Based on the value of D and E, the input distribution γ of Alice and Bob is decided as follows: if D chooses Alice, then the input X of Alice is chosen uniformly at random from the multiset $\{E-a, E-b, E, E, E, E, E+a, E+b\}$, and Bob is given E; if D chooses Bob, γ just swaps the role of Alice and Bob. Clearly, γ is a product distribution conditioned on (D, E). We have that $\zeta = (\gamma, \xi)$ is a mixture of product distributions. Let $\eta = \zeta^n$, and let μ be the distribution obtained by marginalizing $(\mathbf{D}, \mathbf{E})$ from η. Since every joint vector created above is from V_0, μ is a *collapsing distribution* for f (see[17]). Then by Lemma 5.1 and Lemma 5.5

of,[17] we have the following direct sum result:

$$I(\mathbf{X}, \mathbf{Y}; \Pi(\mathbf{X}, \mathbf{Y}) \mid \mathbf{D}, \mathbf{E}) \geq n \cdot \mathrm{CIC}_{\zeta,2/3}(\mathrm{DIST}).$$

We now expand the terms of $\mathrm{CIC}_{\zeta,2/3}(\mathrm{DIST})$. Let $[a, m-a] = \{a, a+1, a+2, \ldots, m-a\}$, U_e be the random number from the uniform distribution on multiset $K \equiv \{e-a, e+a, e, e, e, e, e-b, e+b\}$, and Ψ be the transcript of a communication-optimal randomized protocol for the primitive function DIST. Let $\beta_{e,y}$ be the distribution of $\Psi(e, y)$ and $\beta_e = \frac{1}{8}\sum_{y \in K} \beta_{e,y}$ be the average distribution. The following inequalities follow from Lemma 2.45 and Lemma 2.52 in Bar-Yossef's PhD thesis,[73] the Cauchy-Schwarz inequality, and the triangle inequality.

$$
\begin{aligned}
&\mathrm{CIC}_{\zeta,2/3}(\mathrm{DIST}) \\
=&\frac{1}{2|m-2a|}\sum_{e\in[a,m-a]}[I(U_e;\Psi(U_e,e)) \\
&\qquad + I(U_e;\Psi(e,U_e))] \\
=&\sum_{\substack{e\in[a,m-a]\\ e_j\in K}}\frac{Pr[U_e=e_j]}{2(m-2a)}[KL(\beta_{e,e_j}||\beta_e) \\
&\qquad + KL(\beta_{e_j,e}||\beta_e)] \\
&\text{(Bar-Yossef's Thesis Proposition 2.45)} \\
\geq&\frac{1}{8\ln 2(m-2a)}\sum_{\substack{e\in[a,m-a]\\ j\in\{-a,-b,0,..0,a,b\}}}[h^2(\beta_{e+j,e},\beta_e) \\
&\qquad + h^2(\beta_{e,e+j},\beta_e)] \\
&\text{(Cauchy-Schwarz)} \\
\geq&\frac{1}{16\ln 2(m-2a)}\sum_{e\in[a,m-a]}[(h(\beta_{e,e},\beta_e)+h(\beta_{e,e+a},\beta_e))^2 \\
&+(h(\beta_{e-a,e},\beta_e)+h(\beta_{e,e},\beta_e))^2 \\
&+(h(\beta_{e,e},\beta_e)+h(\beta_{e,e+b},\beta_e))^2 \\
&+(h(\beta_{e-b,e},\beta_e)+h(\beta_{e,e},\beta_e))^2] \\
&\text{(Triangle inequality and Cauchy-Schwarz)} \\
\geq&\frac{C}{m}\sum_{e\in[2a,m-2a]}h^2(\beta_{e,e},\beta_{e+a,e+a})+h^2(\beta_{e,e},\beta_{e+b,e+b}), \qquad (2.6.1)
\end{aligned}
$$

where C is a constant. Now we are able to group the terms. By the Euclidean algorithm,

there exist integers p and q such that $pa + qb = 1$. Let q be such an integer with smallest absolute value. We then have $a/b \leq |q| \leq a$ and $|p| \leq |q|$. Therefore, for any $i \in [4, m/4a-4]$, we can select up to $|p| + |q|$ terms from the range $[4ai - 2a, 4ai + 2a]$ to group in the above equation. Without loss of generality, assume $y > 0$. First, using the Cauchy-Schwarz inequality, we have

$$\begin{aligned}\mathrm{CIC}_{\zeta,2/3}(\mathrm{DIST}) \geq & \frac{C}{4ma} \sum_i \Bigg(\sum_{e \in [4ai-2a, 4ai+2a]} \\ & h(\beta_{e,e}, \beta_{e+a,e+a}) + h(\beta_{e,e}, \beta_{e-a,e-a}) \\ & + h(\beta_{e,e}, \beta_{e+b,e+b}) + h(\beta_{e,e}, \beta_{e-b,e-b}) \Bigg)^2 .\end{aligned}$$

Next, group the terms using the triangle inequality such that whenever combining an a-term $h(\beta_{e,e}, \beta_{e'+a,e'+a})$, follow this by combining $\lfloor q/p \rfloor$ b-terms $h(\beta_{e,e}, \beta_{e''-b,e''-b})$. Therefore, the combined term e' is always guaranteed to be in $[4ai-2a, 4ai+2a]$. There might be a concern that a term is combined twice, namely, that there exist $|q_1| \leq |q_2| \leq |q|$, $|p_1| \leq |p_2| \leq |p|$ and $(q_1, p_1) \neq (q_2, p_2)$ such that $p_1 a + q_1 b = p_2 a + q_2 b$. But this is impossible since $\gcd(a, b) = 1$ and $(p - (p_2 - p_1))a + (q - (q_2 - q_1))b = 1$ contradicts that q has the smallest absolute value. Therefore, we are left with,

$$\mathrm{CIC}_{\zeta,2/3}(\mathrm{DIST}) \geq \frac{C'}{ma} \sum_{\substack{i \in [4, m/4a-4] \\ e = 4ai}} h^2(\beta_{e,e}, \beta_{e+1,e+1}). \tag{2.6.2}$$

Now invoke the Pythagorean lemma of,[17] stating that $h^2(\beta_{e,e}, \beta_{e+1,e+1}) \geq 1/2(h^2(\beta_{e,e}, \beta_{e+1,e}) + h^2(\beta_{e+1,e}, \beta_{e+1,e+1}))$, as well as the correctness of the protocol, stating that the $h^2(\beta_{e,e}, \beta_{e,e+1})$

terms can each be lower bounded by a positive constant. Thus,

$$\mathrm{CIC}_{\zeta,2/3}(\mathrm{DIST}) \geq \frac{C''}{a^2}. \tag{2.6.3}$$

□

The above proof also makes use of the following lemma.

Lemma 2.6.15. Let a, b be two integers such that $\gcd(a, b) = 1$ and $b < a$. Then there exist integers x, y such that $ax + by = 1$. Let y be such an integer with smallest absolute value. Then $b/a \leq |y| \leq a$ and $|x| \leq |y|$.

Proof. To see that $|y| \leq a$, w.l.o.g., we assume $y > a$. Then $y = qa + r$, where $r < a < y$ and $q < y$. Thus $ax + b(qa + r) = 1$, and $(qb + x)a + rb = 1$ contradicts that y has the smallest absolute value. It is clear that $|y| \geq a/b$. It remains to show that $|x| \leq |y|$. This holds since $x = (1 - by)/a$ and $b < a$. □

Now we look at a more general case.

Theorem 2.6.16. Let a, b, c be positive integers such that $c \neq a$ and $c \neq b$. Suppose there exist integers p, q such that $ap + bq = c$. Let q be such an integer with smallest absolute value. Then any randomized algorithm solving (a, b, c)-DIST with probability at least $2/3$ requires $\Omega(n/q^2)$ bits of memory.

Proof. The proof of this theorem is a modification of the proof of the previous proposition. With a similar distribution (replacing 1 with c), we have that Equation (2.6.1) holds. The remaining task is to show a partition scheme of the terms such that $\Theta(m/(|p| + |q|)^2)$ terms

survive. Choose an arbitrary $e \in [4a^2, m - 4a^2]$. W.l.o.g. assume $p > 0$. We can combine the following $|p| + |q|$ terms,

$$
\begin{aligned}
& h^2(\beta_{e,e}, \beta_{e+a,e+a}) + h^2(\beta_{e+a,e+a}, \beta_{e+2a,e+2a}) \\
& \quad + \ldots + \\
& \quad + h^2(\beta_{e+pa-a,e+pa-a}, \beta_{e+pa,e+pa}) \\
& \quad + h^2(\beta_{e+pa,e+pa}, \beta_{e+pa-b,e+pa-b}) \\
& \quad + h^2(\beta_{e+pa-b,e+pa-b}, \beta_{e+pa-2b,e+pa-2b}) \\
& \quad + \ldots + \\
& \quad + h^2(\beta_{e+pa+qb+b,e+pa+qb+b}, \beta_{e+pa+qb,e+pa+qb}) \\
& \geq \frac{1}{|p| + |q|}[\sum_{i=1}^{p} h(\beta_{e+(i-1)a,e+(i-1)a}, \beta_{e+ia,e+ia}) \\
& + \sum_{j=1}^{|q|} h(\beta_{e+pa-(j-1)b,e+pa-(j-1)b}, \beta_{e+pa-jb,e+pa-jb})]^2 \\
& \qquad \text{(Cauchy-Schwarz)} \\
& \geq \frac{h^2(\beta_{e,e}, \beta_{e+1,e+1})}{|p| + |q|} \qquad \text{(Triangle inequality).}
\end{aligned}
$$

Choose an e. This forbids the choice of another e from the above terms. In fact, e uniquely determines two arithmetic progressions: $e, e + a, e + 2a, \ldots, e + pa$ and $e + pa - b, e + pa - 2b \ldots, e + pa + qb$. Now, in order to choose a new e', the value e' cannot be a number from the above two progressions, and the progressions determined by e' cannot share a term with one of e. Therefore, because we chose e, $2(|p| + |q|)$ terms are not available for a new choice of e'. By choosing e from $[4a^2, m - 4a^2]$ one at a time, we can find $(m - 8a^2)/a$ different e

for the purpose of combining terms. Denote the set of these e by S. In summary, we have,

$$\mathrm{CIC}_{\zeta,2/3}(\mathrm{DIST}) \geq \frac{C}{m(|p|+|q|)} \sum_{e \in S} h^2(\beta_{e,e}, \beta_{e+1,e+1}). \tag{2.6.4}$$

Since $h^2(\beta_{e,e}, \beta_{e+1,e+1})$ is bounded below by a constant as indicated in the previous proposition, and since $|S| = m - 8a^2 = \Omega(m)$, we have $\mathrm{CIC}_{\zeta,2/3}(\mathrm{DIST}) = \Omega(1/q^2)$. □

The above lower bound is tight in the sense that there is a protocol that uses $O(n/q^2)$ memory bits and solves (a, b, c)-DIST.

Proposition 2.6.17. Let a, b, c, p, q be integers satisfying the same conditions as in the previous theorem. Then there is a randomized algorithm solving (a, b, c)-DIST with probability at least $2/3$ using $O(n/q^2) \cdot \mathrm{poly}(\log n)$ bits of memory.

Proof. Before the beginning of the stream, we partition $[n]$ into t contiguous pieces each of size n/t, where $t = \widetilde{O}(n/q^2)$, and the $\widetilde{O}$ notation hides logarithmic factors in n. For the i-th piece of the universe, let the corresponding substream restricted to elements in the i-th piece be denoted by S_i. For each S_i, we maintain a counter C_i for which

$$C_i = \sum_{l:h[l]=i} v_l \xi_l = \sum_{x \in \{a,b,1\}} z_{i,x} x$$

where $\xi_l \sim \{-1, +1\}$ are uniform 4-wise independent random bits and $z_{i,x} = \sum_{l:h[l]=i, |v_l|=x} \xi_l$. Let $y_{i,x}$ denote the number of occurrence of $|x|$ in the i-th stream. We have $y_{i,x} \leq n/t$. By standard concentration arguments, we have that with arbitrarily large constant probability, $|z_{i,x}| = \widetilde{O}(\sqrt{y_{i,x}}) = \widetilde{O}(\sqrt{n/t})$. We can select an appropriate $t = \widetilde{O}(n/q^2)$ for which this

implies $|z_{i,x}|$ is at most $|q|/4$. Now the claim is that by reading the value $C_i \mod a$, this is enough to distinguish whether there is a frequency of absolute value "c" in the stream or not. This follows since the sets of values of $C_i \mod a$ in the two cases are disjoint. To see why, note that if $z'_b b = z_b b + c \mod a$, then we have $(z'_b - z_b) - ra = c$. However, since $|z'_b - z_b| < |q|$, by the minimality of $|q|$, this is a contradiction. □

ShortLinearCombination Problem for Multiple of Frequencies

Definition 2.6.18. Let $u = (u_1, u_2, \ldots u_r)$ be a vector in $\mathbb{Z}^r$ for an integer r. Let $d > 0$ be an integer not in the vector u. A stream S with frequency vector v is given to an algorithm, where v is promised to be from $V_0 = \{u_1, u_2, \ldots, u_r, 0\}^n$ or $V_1 = \{\text{EMB}(v, i, e) \mid v \in V_0, i \in [n], e \in \{-d, d\}\}$. The (u, d)-DIST problem is to distinguish whether $v \in V_0$ or $v \in V_1$.

Theorem 2.6.19. Let $u = (a_1, a_2, \ldots, a_r)$ be a set of integers and $d > 0$ be a positive integer such that $d = q_1 a_1 + q_2 a_2 + \ldots q_r a_r$ where $q_1, q_2, \ldots q_r$ are integers. These integers are choosen such that $q = \sum_i |q_i|$ is minimal. Then any randomized algorithm solving (u, d)-DIST with probability at least $2/3$ requires $\Omega(n/q^2)$ bits of memory.

Proof. The proof of the lower bound is a straightforward extension of the previous argument for three frequencies. Let $a = \max(a_1, a_2, \ldots, a_r)$. The number of combined terms is $m/(|q_1| + |q_2| + \ldots |q_r|)$. Therefore, the lower bound is $\Omega(n/q^2)$. The upper bound is a also a straightforward extension of the three frequency case. □

Chapter 3

Nearly Optimal Characterization of Streaming Symmetric Norms

This chapter is based on BBCKY (STOC 2017).[21]

3.1 Background

The study of norms on data streams has a rich history, and in particular has driven much of the fantastic development of streaming algorithms, see e.g.[7,14–16] A *data stream* is a sequence of additive ± 1 updates that accumulate on the coordinates of an n-dimensional vector v, and a streaming algorithm reads the sequence of updates and computes some function of v. This is known as the turnstile model, and for simplicity we assume that $|v_i| \leq \text{poly}(n)$, for all $i \in [n]$. Despite plenty of work, it is still an open problem to design

a generic streaming algorithm for approximating norms. Although very challenging, it may not be too much to ask for. In fact, several existing methods, including the Indyk-Woodruff sketch,[15,74] yield so-called "universal sketches" that can be used to approximate whole classes of streaming problems at once. So we ask, is there a *generic method* that can approximate any desired norm of a stream with near-optimal space complexity? Second, is there a *universal sketch* whose single evaluation on a vector (say on a stream) suffices to approximate every norm in a wide class? While several powerful upper and lower bound techniques have been developed, including embeddings, heavy-hitters, and reductions from Communication Complexity, it is not apparent how they can be applied to an entirely new norm, see also Open Problems 5 (Sketchable Distances) and 30 (Universal Sketching) in the list.[75]

This is a real challenge for at least two reasons. First, we lack a generic framework for embeddings. Even when it is possible to embed into an easy-to-handle space, a new embedding must be constructed and applied to the input stream for each norm. Second, current techniques, heavy-hitters included, have been confined to norms with additive structure. Nearly all of the norms considered so far decompose, on some level, into a sum of independent quantities, and this fact is heavily exploited in the design of algorithms and lower bounds. Examples include l_p norms (see references in Section 3.1.3), the entropy norm,[76–78] and cascaded l_p norms.[79,80] Abandoning our reliance on additive decomposability has been a major bottleneck en route to a broader characterization of norms.

We overcome this barrier in the setting of symmetric norms, see e.g. [81, Chapter IV]. A norm $l : \mathbb{R}^n \to \mathbb{R}$ is called *symmetric* if, for all $x \in \mathbb{R}^n$ and every $n \times n$ permutation

matrix P, it satisfies $l(Px) = l(x)$ and also $l(|x|) = l(x)$, where $|x|$ is the coordinate-wise absolute value of x. It is a partial answer to the question above, as we design a generic algorithm for symmetric norms and it is based on a universal sketch. Specifically, for every $s > 0$, there is a single sketch of size $s \cdot \text{poly}(\log(n)/\epsilon)$, that yields a $(1 \pm \epsilon)$-approximation[1] for every symmetric norm whose streaming space complexity is at most s. In fact, we show that the streaming space complexity of a symmetric norm is determined by the norm's measure-concentration characteristics. To be precise, let $X \in \mathbb{R}^n$ be uniformly distributed on S^{n-1}, the l_2 unit sphere. The median of a symmetric norm l is the (unique!) value M_l such that $\Pr[l(X) \geq \text{M}_l] \geq 1/2$ and $\Pr[l(X) \leq \text{M}_l] \geq 1/2$. Similarly, $\mathfrak{b}_l$ denotes the maximum value of $l(x)$ over $x \in S^{n-1}$. We call the ratio

$$\text{mc}(l) := \mathfrak{b}_l \,/\, \text{M}_l$$

the *modulus of concentration* of the norm l. Our results show that this modulus of concentration is crucial in determining the streaming space complexity of any symmetric norm. This quantity governs many phenomena in high-dimensional spaces, for example, it appears in large-deviation bounds and the critical dimension in Dvoretzky's Theorem is $n/\,\text{mc}(l)^2$, see e.g.[82,83]

Symmetric norms clearly include the l_p and entropy norms, and we present fresh examples with heretofore unknown streaming space complexity, like the top-k norm, Q norms,

[1] We state the approximation ratio in one of two standard ways. A D-approximation, $D \geq 1$, to $l(v)$ is a value $\widehat{l}$ such that $l(v) \leq \widehat{l} \leq Dl(v)$. When D is very close to one, it is more convenient to consider a $(1 \pm \epsilon)$-approximation, $0 \leq \epsilon < 1/2$, which is defined as $(1-\epsilon)l(v) \leq \widehat{l} \leq (1+\epsilon)l(v)$ and corresponds to a D-approximation for $D = \frac{1+\epsilon}{1-\epsilon}$.

and Q' norms, later on. Although matrix norms are generally not symmetric, our results immediately imply lower bounds for unitarily invariant matrix norms, for example the Ky Fan norms, by restricting attention to diagonal matrices.

One well-studied family of symmetric norms is that of l_p norms on $\mathbb{R}^n$, defined as $l_p(x) := (\sum_{i=1}^n |x_i|^p)^{1/p}$. For $1 \leq p \leq 2$, the maximum value of $l_p(x)$ over $x \in S^{n-1}$ is $\mathfrak{b}_{l_p} = n^{1/p-1/2}$ and concentrates at $\mathrm{M}_{l_p} = \Theta(n^{1/p-1/2})$, so the modulus of concentration is $\mathrm{mc}(l_p) = O(1)$. For $p > 2$, the maximum is $\mathfrak{b}_{l_p} = 1$ but again concentrates at $\mathrm{M}_{l_p} = \Theta(n^{1/p-1/2})$, hence $\mathrm{mc}(l_p) = \Theta(n^{1/2-1/p})$. Recall that the streaming space complexity for a $(1 \pm 1/10)$-approximation of l_p is $\Theta(\log n)$, when $p \leq 2$,[84] and is $\Theta(n^{1-2/p} \log n)$ when $p > 2$[68,71] (the constant $1/10$ here is arbitrary). Thus for all values of $p \geq 1$, the space complexity of computing a $(1 \pm 1/10)$-approximation to l_p is $\Theta(\mathrm{mc}(l_p)^2 \log n)$. Our main result recovers this fact up to a polylog$n$ factor.

But, the modulus of concentration cannot be the whole story for streaming algorithms. It expresses an average behavior of the norm on $\mathbb{R}^n$, and even if the norm is well-behaved on average, like l_1 for example, it is possible that a more difficult norm is concealed in a lower-dimensional subspace. One example of this is $l(x) := \max\{l_\infty(x), l_1(x)/\sqrt{n}\}$ on $\mathbb{R}^n$, which has $\mathrm{mc}(l) = O(1)$. However, when x has fewer than $\sqrt{n}$ nonzero coordinates, $l(x) = l_\infty(x)$, which is just a lower-dimensional copy of l_∞ and implies, by,[14] an $\Omega(\sqrt{n})$ space lower bound for l. In order for the modulus of concentration to have any connection with streaming space complexity, we have to close this gap.

Notice that, for every $k \leq n$, the norm l induces a norm $l^{(k)}$ on $\mathbb{R}^k$ by setting

$$l^{(k)}((x_1, x_2, \ldots, x_k)) := l((x_1, \ldots, x_k, 0, \ldots, 0)).$$

Of course, because of the permutation symmetry we could have chosen any set of $n - k$ coordinates to be the zeros. As the examples above show, the modulus of concentration of $l^{(k)}$ may vary with k. However, any streaming approximation algorithm for l is also trivially a streaming approximation algorithm for $l^{(k)}$. We therefore define the *maximum modulus of concentration* of the norm l as

$$\mathrm{mmc}(l) := \max_{k \leq n} \mathrm{mc}(l^{(k)}) = \max_{k \leq n} \frac{\mathfrak{b}_{l^{(k)}}}{\mathrm{M}_{l^{(k)}}}.$$

Our main result is that this quantity characterizes the streaming space complexity of every symmetric norm l.

3.1.1 Our Results

Quite surprisingly, for every symmetric norm l on $\mathbb{R}^n$, the optimal space complexity of a streaming algorithm that gives a $(1 \pm \epsilon)$-approximation for l is $\mathrm{mmc}(l)^2 \cdot \mathrm{poly}(\log(n)/\epsilon)$. This characterization tells us in particular whether a given symmetric norm admits a poly-logarithmic space approximation or requires polynomial space.

Theorem 3.1.1 (Main Theorem)**.** Let l be a symmetric norm on $\mathbb{R}^n$. For every $\epsilon > 0$, there is a one-pass streaming algorithm that on an input stream vector $v \in \mathbb{R}^n$ computes, with probability at least 0.99, a $(1 \pm \epsilon)$-approximation to $l(v)$, and uses $\mathrm{mmc}(l)^2 \cdot \mathrm{poly}(\log(n)/\epsilon)$

bits of space.

Theorem 3.1.2 (Lower Bound)**.** Let l be a symmetric norm on $\mathbb{R}^n$. Any turnstile streaming algorithm that outputs, with probability at least 0.99, a $(1 \pm 1/6)$-approximation for $l(\cdot)$ must use $\Omega(\text{mmc}(l)^2)$ bits of space in the worst case.

For the coarser D-approximation, where $D > 1.1$ and can grow with n, in Theorem 3.1.3 we build upon the algorithm of Theorem 3.1.1 trading the larger approximation ratio for a $1/D^2$ multiplicative decrease in storage. It turns out that the quadratic dependence on D is the best possible; we prove the matching lower bound in Theorem 3.1.4.

Theorem 3.1.3. Let l be a symmetric norm on $\mathbb{R}^n$. For every $1.1 \leq D \leq \text{mmc}(l)$ there is a one-pass streaming algorithm that on input stream vector $v \in \mathbb{R}^n$ computes, with probability at least 0.99, a D-approximation to $l(v)$ and uses $(\text{mmc}(l)^2/D^2) \cdot \text{poly}(\log n)$ bits of space.

Theorem 3.1.4. Let l be a symmetric norm on $\mathbb{R}^n$. Any turnstile streaming algorithm that outputs, with probability at least 0.99, a D-approximation for $l(\cdot)$ must use $\Omega(\text{mmc}(l)^2/D^2)$ bits of space in the worst case.

We prove the upper bound theorems in Sections 3.3 and 3.5, respectively, with some details in the full version of this paper.). The lower bounds both appear in Section 3.4. To our knowledge, this is the first application of measure concentration to streaming algorithms (Chernoff and Hoeffding bounds aside). The geometric and analytical properties of high-dimensional normed spaces have become well understood over decades of research. We hope that more tools from that field can be brought to bear on these intriguing streaming problems, see Section 3.8 for promising directions for further work.

Applications and Examples. Section 3.6 describes some applications of our results. One application is to a class of norms called Q' norms,[81] which includes the l_p norms for $1 \leq p \leq 2$, among others. Q' norms are just the dual norms to Q norms (shorthand for quadratic), which in turn are norms of the form $l(x) = \Phi(x^2)^{1/2}$, for some symmetric norm Φ, where x^2 denotes coordinate-wise squaring of x. We study these norms in Section 3.6.2. The upshot is that every Q' norm l' has $\text{mmc}(l') = O(\log n)$, and thus can be computed by a streaming algorithm using polylogarithmic space. Several Q' norms have been proposed as regularizers for sparse recovery problems in Machine Learning. One such norm is the k-support norm,[85] which is more conveniently described via its unit ball $C_k = \text{conv}\{x \in \mathbb{R}^n : |\text{supp}(x)| \leq k \text{ and } l_2(x) \leq 1\}$. It is not readily apparent how to design a specialized streaming algorithm for this norm, but we obtain such an explicit algorithm, for every k, as a special case of Q' norms. Another example is the box-Θ norm,[86] where given $0 < a < b \leq c$, we let $\Theta := \{\theta \in [a, b]^n : l_1(\theta) \leq c\}$, and define the box-$\Theta$ norm as

$$l_\Theta(x) := \min_{\theta \in \Theta} \Big(\sum_{i=1}^{n} x_i^2/\theta_i \Big)^{1/2}, \qquad \text{and its dual norm is} \quad l'_\Theta(x) := \max_{\theta \in \Theta} \Big(\sum_{i=1}^{n} \theta_i x_i^2 \Big)^{1/2}.$$

It's easy to see that every box-Θ norm is a Q' norm, and therefore has polylogarithmic streaming space complexity. To the best of our knowledge, there is no other technique that can approximate these norms on a streaming vector.

Our results also apply to what we shall call the *top-k norm*. Denoted as $\Phi_k(x)$, it is defined as the sum of the k largest coordinates of $|x|$.[81] This norm is a special case of the Ky Fan k-norm and is sometimes studied as a toy example to understand regularization

of the Ky Fan norms.[87] We show in Section 3.6.1 that $\text{mmc}(\Phi_k) = \widetilde{\Theta}(\sqrt{n/k})$, so when k is large, for example linear in n, the top-k norm of a stream vector can be approximated in only polylogarithmic space. We are aware of no other streaming algorithms that can approximate this norm, as ours does.

3.1.2 Overview of Techniques

Upper Bound. Our algorithm for Theorem 3.1.1 uses a linear sketch in the style of Indyk and Woodruff's sketch for large frequency moments,[15] but the size of the sketch is calibrated by $\text{mmc}(l)$. The algorithm is presented in Section 3.3, with some details presented in the full version. This is a surprising application of the Indyk-Woodruff sketching technique, as all previous applications of this method are to computing functions with an additive structure $\sum_{i=1}^{n} f(v_i)$. In these settings, the Indyk-Woodruff algorithm can be viewed as performing Importance Sampling of the summands of the target function $\sum_{i=1}^{n} f(v_i)$. However, a symmetric norm l need not have an explicit mathematical formula, let alone be decomposable as a sum, and we thus need a different way to identify the "important" coordinates, which informally means that zeroing these coordinates would introduce too much error to $l(v)$. At a high level, our analysis makes two major contributions. The first is to provide an explicit criterion for importance, and the second is to reveal that inside this importance criterion, the most crucial quantity is the maximum modulus of concentration $\text{mmc}(l)$. A more detailed outline of the analysis follows, omitting constants and dependence on ϵ.

First, we imagine rounding each coordinate of the streaming vector v to a power of $\alpha = 1 + 1/\text{polylog}(n)$, which can be seen to have negligible effect using basic properties

of symmetric norms. Moreover, since the norm is symmetric, it suffices to know only the number of coordinates, b_i, at each "level" α^i. By our choice of α, there are only polylog(n) levels, so we can represent the rounded vector succinctly. Recovering the rounded vector exactly would require linear storage, so we use the Indyk-Woodruff sampling technique to approximate the vector.

The Indyk-Woodruff procedure approximates each b_i by sampling each coordinate i of the vector v with probability polylog$(n)/b_i$, and then in the sampled vector (which is expected to have polylog(n) coordinates of level i whenever $b_i \neq 0$), the algorithm identifies l_2-heavy-hitters. If the coordinates of level i are l_2-heavy-hitters in the sampled vector (they are in the same level and thus have about the same value), then we get a good estimate of b_i; it's not as simple as counting them and scaling inversely to the sampling probability, but that is the right idea. If the coordinates are not l_2-heavy-hitters, then we get no estimate for b_i, and must assume it is 0. We show that if we parameterize the sketch according to mmc$(l)^2$, then we get approximations to all the "important" levels, which is sufficient to accurately recover $l(v)$.

Lower Bound. The lower bound of Theorem 3.1.2 is proved using a reduction from the Communication Complexity of multiparty set-disjointness, and concentration of measure of the norm l again plays a key role. In the disjointness setting, each of t players is given a subset of $[n]$, and their task is to determine whether the sets are mutually disjoint or are "uniquely" intersecting. Instead of the standard reduction, where each player places in the stream one update to v_i for every element $i \in [n]$ in the set he holds, in our reduction,

each player $j \in [t]$ adds to the stream a vector $w^{(i,j)} \in \mathbb{R}^n$ whenever element i is in his set. Each vector $w^{(i,j)}$ is random but the entire collection of vectors is designed so that the resulting stream vector is, roughly, a uniformly random vector on a "disjoint" instance, and a vector maximizing the norm on an "intersecting" instance. For these two cases to be well-separated, we must choose the number of players t to be large enough. By applying concentration of measure, we show that $t = O(\sqrt{n}/\operatorname{mmc}(l))$ players suffice, and, by known communication bounds for disjointness,[17,70,71] this leads to an $\Omega(n/t^2) = \Omega(\operatorname{mmc}(l)^2)$ storage lower bound for every algorithm approximating the norm l to within $1 \pm 1/6$ (the constant $1/6$ is arbitrary). Extending the lower bound to a D-approximation, for D bounded away from 1, can be accomplished with the same reduction using $t = O(D\sqrt{n}/\operatorname{mmc}(l))$ players instead, which yields Theorem 3.1.4. The proofs of both lower bound theorems can be found in Section 3.4.

Optimal Tradeoff. For the D-approximation algorithm, Theorem 3.1.3, the idea is to define, given a norm l, a new symmetric function $l_{(D)} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ such that $l(x) \leq l_{(D)}(x) \leq Dl(x)$. Even though $l_{(D)}$ is not a norm, we can still define $\operatorname{mmc}(l_{(D)})$, which is bounded as $\widetilde{O}(\operatorname{mmc}(l)/D)$. The approximation comes by using our main algorithm to get a 1.1-approximation to $l_{(D)}(v)$, which translates into a $2D$-approximation of $l(v)$. The definition of $l_{(D)}$ and its analysis are presented in Section 3.5.

3.1.3 Related Work

There has been extensive work on computing norms, and related functions, in the sketching and streaming models. Most recently, Andoni, Krauthgamer, and Razenshteyn[60] have shown that a normed space $(\mathbb{R}^n, l)$ embeds linearly into $l_{0.99}$ with distortion $D > 1$ if and only if this normed space admits distance estimation sketching with approximation $\Theta(D)$ and sketch size $O(1)$ bits. Thus, they characterize sketching of a general norm by its embeddability. In comparison, our characterization applies only to symmetric norms, but we consider streaming (not sketching) algorithms, which in Theorem 3.1.1 means a stronger consequence, and in Theorem 3.1.2 means a stronger assumption. And perhaps more importantly, our results achieve $(1+\epsilon)$-approximation, while their algorithm achieves approximation proportional to D (though their lower bound shows a linear tradeoff with sketch size).

Another important tool that may seem relevant is that every turnstile streaming algorithm can be replaced by a linear sketch, as shown by Li, Nguyen, and Woodruff.[61] However, this transformation does not make it easy to determine the streaming complexity of a given symmetric norm l, because it is not easy to design a linear sketch for l.

There are other generic streaming algorithms that provide approximation guarantees for an entire class of functions of the form $\sum_{i=1}^{n} f(v_i)$, where f is some nonnegative function.[18,66] If one has a so-called f-heavy-hitters algorithm that identifies every coordinate i accounting for half of the total sum, i.e., $f(v_i) \geq \sum_{j \neq i} f(v_j)$ and moreover approximates this $f(v_i)$, then one can also approximate the sum $\sum_{i=1}^{n} f(v_i)$, incurring only an $O(\log n)$ factor

overhead on top of the f-heavy-hitters algorithm's storage. For a large class of functions f, including monotone functions, computing f-heavy-hitters can be reduced to computing l_2-heavy-hitters in several random sub-streams[20,74] or even just random sampling.[19] Universality falls out as a side-effect of the design of the algorithm — the only dependence on f is through the number of sub-streams, which determines the sketch size, up to a polylog(n) factor. Therefore, any two functions that lead to the same sketch size, in fact, use the exact same sketch.

Finally, we should mention there is a very long line of results on estimating l_p norms (also called frequency moments) in a data stream, including designing small-space algorithms[14–16,66,84,88–93] and proving space lower bounds.[17,68,70,71,94,95] This list omits improvements of the runtime of update and output procedures, and devising extensions like l_p sampling.

3.2 Preliminaries

An important unit vector for us is $\xi^{(n')} := \frac{1}{\sqrt{n'}}(1,1,1,\ldots 1,0,\ldots,0) \in \mathbb{R}^n$, for any $n' \leq n$, which has n' nonzero coordinates. We abuse the notation to write $\xi^{(n')} \in \mathbb{R}^{n'}$ by removing zero coordinates, and vice-versa by appending zeros. Let us record some basic facts about symmetric norms.

Lemma 3.2.1 (Monotonicity of Symmetric Norms, see e.g. Proposition IV.1.1 in[81])**.** If $l(\cdot)$ is a symmetric norm and $x, y \in \mathbb{R}^n$ satisfy that for all i, $|x_i| \leq |y_i|$, then $l(x) \leq l(y)$.

Without loss of generality, we assume that our norms are normalized on the standard

basis, i.e., $l(e_i) = 1$. Recall that the *dual* of a norm $l : \mathbb{R}^n \to \mathbb{R}$ is the norm $l' : \mathbb{R}^n \to \mathbb{R}$ given by $l'(x) := \sup\{\frac{|\langle x,y\rangle|}{l(y)} : y \neq 0\}$. For the following facts see, e.g., [82, Sections 3.1.2 and 4.5].

Fact 3.2.2. For all $x \in \mathbb{R}^n$, $l_\infty(x) \leq l(x) \leq l_1(x)$.

Fact 3.2.3. Let $a, b > 0$ be such that, for all $x \in \mathbb{R}^n$, $a^{-1}\, l_2(x) \leq l(x) \leq b\, l_2(x)$. Then, for all $x \in \mathbb{R}^n$, $b^{-1}\ l_2(x) \leq l'(x) \leq a\, l_2(x)$.

Fact 3.2.4. $\mathrm{M}_l\, \mathrm{M}_{l'} \geq 1$.

We restrict attention to vectors v whose coordinates are in the range $\{-m, \ldots, m\}$, for $m = \mathrm{poly}(n)$, so $\log m = O(\log n)$. Our results still apply when m is larger but one must replace $\log n$ factors with $\log m$ factors.

Last, we must be precise about the model of computation, because we do not have a mathematical formula for the norm. Our algorithm will rely on evaluating the norm on a vector that is derived from a sketch of the stream. Every coordinate of this vector should be easy to recover from the sketch, but the vector need not be written explicitly, to avoid $\Omega(n)$ storage. To accomodate this, we make the assumption that our algorithm has access to an oracle `NORM` that computes $l(v)$ using queries to the coordinates of v, i.e., our algorithm must provide query access to any coordinate v_i.

3.3 An Algorithm for Symmetric Norms

In this section we prove Theorem 3.1.1, which shows that a symmetric norm can be approximated in the turnstile streaming model using one pass and $O(\mathrm{mmc}(l)^2 \mathrm{poly}(1/\epsilon \cdot$

$\log n)$) bits of memory. The Algorithm 3, uses a subroutine called `Level1`, whose full description appears in the full version. The rest of this section considers a given symmetric norm l on $\mathbb{R}^n$ and a desired accuracy parameter $0 < \epsilon < 1$. Let the two parameters $\alpha > 1$ and $0 < \beta \leq 1$ be determined later, possibly depending on n, ϵ and $\text{mmc}(l)$. We assume $\text{mmc}(l) \leq \gamma\sqrt{n}$, for some sufficiently small constant $0 < \gamma \ll 1/2$, since otherwise the lower bound given in Theorem 3.1.2 implies that linear memory is necessary to approximate this norm with a streaming algorithm.

3.3.1 Level Vectors and Important Levels

Definition 3.3.1 (Important Levels)**.** For $v \in \mathbb{R}^n$, define *level* i as $B_i := \{j \in [n] : \alpha^{i-1} \leq |v_j| < \alpha^i\}$, and denote its size by $b_i := |B_i|$. We say that level i is β-*important* if

$$b_i > \beta \sum_{j>i} b_j; \quad \text{and} \quad b_i \alpha^{2i} \geq \beta \sum_{j \leq i} b_j \alpha^{2j}.$$

Recall from Section 3.2 that we restrict attention to vectors v whose coordinates are in the range $\{-m, \ldots, m\}$, for $m = \text{poly}(n)$. This assumption implies that the number of non-zero b_i's is at most $t = O(\log_\alpha n)$. And if we normalize v to a unit vector in l_2-norm, then every non-zero coordinate has absolute value at least $1/\text{poly}(n)$.

We will rely on the next theorem, which shows a streaming algorithm recovers all the important b_i's. Its proof appears in the full version.

Theorem 3.3.2. For every $\epsilon > 0$, there is a one-pass streaming algorithm `Level1` that given an input stream and parameters $\alpha' = 1 + \gamma > 1$ and $0 < \beta \leq 1$, outputs $\{\widehat{b}_i\}$ for base

$\alpha = 1 + O(\gamma)$, such that with probability $1 - O(1/\text{poly}(n))$, for all i,

- $\widehat{b}_i \leq b_i$; and
- if level i is β-important, then $\widehat{b}_i \geq (1-\epsilon)b_i$.

This algorithm uses $O(\gamma^{-5}\epsilon^{-2}\beta^{-1}\log^{12} n)$ bits of space.

To state and analyze our algorithm for approximating $l(v)$, we introduce the following notation. Later, we shall omit (v) from the notation, as it is clear from the context.

Definition 3.3.3 (Level Vectors and Buckets)**.** Define the *level vector* for $v \in \mathbb{R}^n$ with integer coordinates to be

$$V(v) := (\underbrace{\alpha^1, \ldots, \alpha^1}_{b_1 \text{ times}}, \underbrace{\alpha^2, \ldots, \alpha^2}_{b_2 \text{ times}}, \ldots, \underbrace{\alpha^t, \ldots, \alpha^t}_{b_t \text{ times}}, 0, \ldots, 0) \in \mathbb{R}^n;$$

and define the *i-th bucket* of $V(v)$ to be

$$V_i(v) := (\underbrace{0, \ldots, 0}_{b_1+b_2+\ldots+b_{i-1} \text{ times}}, \underbrace{\alpha^i, \ldots, \alpha^i}_{b_i \text{ times}}, \underbrace{0, \ldots, 0}_{b_{i+1}+b_{i+2}\ldots b_t \text{ times}}, 0, \ldots, 0) \in \mathbb{R}^n.$$

Let $\widehat{V}(v)$ and $\widehat{V}_i(v)$ be defined similarly for the approximated values $\{\widehat{b}_i\}$. We denote $V(v)\backslash V_i(v)$ as the vector with the i-th bucket replaced by 0; and denote $V(v)\backslash V_i(v) \cup \widehat{V}_i(v)$ as the vector by replacing the whole i-th bucket with $\widehat{V}_i(v)$, i.e.,

$$V(v)\backslash V_i(v) \cup \widehat{V}_i(v) := (\underbrace{\alpha^1, \ldots, \alpha^1}_{b_1 \text{ times}}, \ldots, \underbrace{\alpha^i, \ldots, \alpha^i}_{\widehat{b}_i \text{ times}}, \ldots, \underbrace{\alpha^t, \ldots, \alpha^t}_{b_t \text{ times}}, 0, \ldots, 0) \in \mathbb{R}^n.$$

3.3.2 Approximated Levels Provide a Good Approximation

We first show the level vector V can be used to approximate $l(v)$, if we choose a base $\alpha := 1 + O(\epsilon)$.

Proposition 3.3.4. For all $v \in \mathbb{R}^n$, $l(V(v))/\alpha \leq l(v) \leq l(V(v))$.

Proof. Follows directly from the monotonicity of symmetric norms (Lemma 3.2.1). □

The next key lemma shows that $l(\widehat{V})$ is a good approximation to $l(V)$.

Lemma 3.3.5 (Bucket Approximation)**.** For every level i, if $\widehat{b}_i \leq b_i$, then $l(V \backslash V_i \cup \widehat{V}_i) \leq l(V)$; and if $\widehat{b}_i \geq (1-\epsilon)b_i$, then $l(V \backslash V_i \cup \widehat{V}_i) \geq (1-\epsilon)l(V)$.

Proof. The upper bound follows immediately from the monotonicity of norms. We will prove the lower bound as follows. Let us take the vector

$$\widehat{V}_i := (\underbrace{0, 0, \ldots 0,}_{b_1+b_2+\ldots+b_{i-1} \text{ times}} \underbrace{\alpha^i, \ldots \alpha^i}_{\widehat{b}_i \text{ times}}, 0, \ldots, 0).$$

Let us also define $W := V - V_i$. Note that $W + \widehat{V}_i$ is a permutation of a vector $V \backslash V_i \cup \widehat{V}_i$. We will prove that, under assumptions of the lemma, $l(W + \widehat{V}_i) \geq (\widehat{b}_i / b_i) l(V)$.

For a vector $v \in \mathbb{R}^n$ and a permultation $\pi \in \Sigma_n$, we denote $\pi(v)$ a vector in $\mathbb{R}^n$ such that $\pi(v)_i := v_{\pi(i)}$. Since the norm l is symmetric, we have that $l(v) = l(\pi(v))$. Consider a set of permultations S, consisting of all permutations that are cyclic shifts over the non-zero coordinates of V_i, and do not move any other coordinates. That is, there is exactly b_i permultations in S, and for every $\pi \in S$, we have $\pi(W) = W$. By the construction of the

set S, we have,

$$\sum_{\pi \in S} \pi(\widehat{V}_i) = \widehat{b}_i V_i$$

and therefore $\sum_{\pi \in S} \pi(W + \widehat{V}_i) = \widehat{b}_i V_i + b_i W$. As vectors V_i and W have disjoint support, by monotonicity of the norm l with respect to each coordinates we can deduce $l(\widehat{b}_i V_i + b_i W) \geq l(\widehat{b}_i(V_i + W))$. By plugging those together,

$$\begin{aligned} \widehat{b}_i l(V_i + W) \leq l(\widehat{b}_i V_i + b_i W) &= l\left(\sum_{\pi \in S} \pi(\widehat{V}_i + W)\right) \\ &\leq \sum_{\pi \in S} l\left(\pi(\widehat{V}_i + W)\right) = b_i l(\widehat{V}_i + W) \end{aligned} \tag{3.3.1}$$

where the last equality follows from the fact that l is symmetric and $|S| = b_i$. Hence, $l(\widehat{V}_i + W) \geq \frac{\widehat{b}_i}{b_i} l(V) \geq (1 - \epsilon) l(V)$, as desired. □

3.3.3 Contributing Levels and Important Levels

Definition 3.3.6 (Contributing Levels)**.** Level i is called *β-contributing* if $l(V_i) \geq \beta\, l(V)$.

Lemma 3.3.7. Let V' be the vector obtained from V by removing all levels that are not β-contributing. Then $(1 - O(\log_\alpha n) \cdot \beta) l(V) \leq l(V') \leq l(V)$.

Proof. Let $i_1, \ldots, i_k$ be the levels that are not β-contributing. Then by the triangle inequality,

$$l(V) \geq l(V) - l(V_{i_1}) - \ldots - l(V_{i_k}) \geq (1 - k\beta) l(V).$$

The proof follows by bounding k by $t = O(\log_\alpha n)$, which is the total number of non-zero

b_i's. □

The following lemma and Lemma 3.3.15 show together that that every β-contributing level is also β'-important for a suitable β' that depends on $\mathrm{mmc}(l)$.

Lemma 3.3.8. If level i is β-contributing, then $b_i \geq \frac{\lambda\beta^2}{\mathrm{mmc}(l)^2 \log^2 n} \cdot \sum_{j>i} b_j$ for some absolute constant $\lambda > 0$.

We present the following concentration of measure results for the proof of this lemma,

Lemma 3.3.9. For every norm l on $\mathbb{R}^n$, if $x \in S^{n-1}$ is drawn uniformly at random according to Haar measure on the sphere, then

$$\Pr(|l(x) - \mathrm{M}_l| > \frac{2\,\mathfrak{b}_l}{\sqrt{n}}) < \frac{1}{3}$$

Lemma 3.3.10. For every $n > 0$, there is a vector $x \in S^{n-1}$ satisfying

1. $|l_\infty(x) - \mathrm{M}_{l_\infty^{(n)}}| \leq 2/\sqrt{n}$,
2. $|l(x) - \mathrm{M}_{l^{(n)}}| \leq 2\ \mathfrak{b}_{l^{(n)}}/\sqrt{n}$, and
3. $|\{i : |x_i| > \frac{1}{K\sqrt{n}}\}| > \frac{n}{2}$ for some universal constant K.

We prove these lemmas using Levy's isoperimetric inequality, see e.g. [82, Section 2.3].

Theorem 3.3.11 (Levy's Isoperimetric Inequality)**.** For a continuous function $f : S^{n-1} \to \mathbb{R}$, let M_f be the median of f, i.e., $\mu(\{x : f(x) \leq \mathrm{M}_f\}) \geq 1/2$ and $\mu(\{x : f(x) \geq \mathrm{M}_f\}) \geq 1/2$, where $\mu(\cdot)$ is the Haar probability measure on the unit sphere S^{n-1}. Then $\mu(\{x : f(x) = \mathrm{M}_f\}_\epsilon) \geq 1 - \sqrt{\pi/2}e^{-\epsilon^2 n/2}$, where for a set $A \subset S^{n-1}$ we denote $A_\epsilon := \{x : l_2(x, A) \leq \epsilon\}$ and $l_2(x, A) := \inf_{y \in A} ||x - y||_2$.

Proof of Lemma 3.3.9. By applying Theorem 3.3.11, for random x distributed according to the Haar measure on the l_2-sphere, with probability at least $1-\sqrt{\pi/2}e^{-2} > \frac{2}{3}$ there is some $y \in S^{n-1}$, such that $\|x-y\|_2 \leq \frac{2}{\sqrt{n}}$ and $l(y) = \mathrm{M}_l$. We know that norm l is $\mathfrak{b}_l$-Lipschitz with respect to $\|\cdot\|_2$, and as such

$$|l(x) - \mathrm{M}_l| = |l(x) - l(y)| \leq l(x-y) \leq \mathfrak{b}_l \|x-y\| \leq \frac{2\,\mathfrak{b}_l}{\sqrt{n}} \qquad \square$$

Proof of Lemma 3.3.10. Consider x drawn uniformly at random from a unit sphere. According to Lemma 3.3.9, we have $\Pr(|l_\infty(x) - \mathrm{M}_{l_\infty^{(n)}}| > 2/\sqrt{n}) < \frac{1}{3}$ and $\Pr(|l(x) - \mathrm{M}_l| > 2\,\mathfrak{b}_l/\sqrt{n}) < \frac{1}{3}$.

Let us define $\tau(x,t) := |\{i : |x_i| < t\}|$. We need to show that for some universal constant K, with probability larger than $\frac{2}{3}$ over a choice of x, we have $\tau(x, \frac{1}{K\sqrt{n}}) < \frac{n}{2}$.

Indeed, consider random vector $z \in \mathbb{R}^n$, such that all coordinates z_i are independent standard normal random variables. It is well known, that $\frac{z}{\|z\|_2}$ is distributed uniformly over a sphere, and therefore has the same distribution as x. There is a universal constant K_1 such that $\Pr(\|z\|_2 > K_1\sqrt{n}) < \frac{1}{6}$, and similarly, there is a constant K_2, such that $\Pr(|z_i| < \frac{1}{K_2}) < \frac{1}{12}$. Therefore, by Markov bound we have $\Pr(\tau(z, \frac{1}{K_2}) > \frac{n}{2}) < \frac{1}{6}$. Using union bound, with probability larger than $\frac{2}{3}$ it holds simultanously that $\|z\|_2 \leq K_1\sqrt{n}$ and $\tau(z, \frac{1}{K_2}) < \frac{n}{2}$, in which case $\tau(z/\|z\|_2, \frac{1}{K_1K_2\sqrt{n}}) < \frac{n}{2}$.

Finally, by union bound, a random vector x satisfies all of the conditions in the statement of the lemma with positive probability. $\square$

We now prove that the norm l of the (normalized) all-ones vector $\xi^{(n)}$ is closely related to the median of the norm. This all-ones vector is useful because it can be easily related to a single level of V.

Lemma 3.3.12 (Flat Median Lemma)**.** Let $l : \mathbb{R}^n \to \mathbb{R}$ be a symmetric norm. Then

$$\lambda_1 \, \mathrm{M}_l \, / \sqrt{\log n} \leq l(\xi^{(n)}) \leq \lambda_2 \, \mathrm{M}_l,$$

where $\lambda_1, \lambda_2 > 0$ are absolute constants.

Note that the first inequality is tight for l_∞. To prove this lemma, we will need the following well-known fact, see e.g.[82]

Fact 3.3.13. There are absolute constants $0 < \gamma_1 \leq \gamma_2$ such that for every integer $n \geq 1$,

$$\gamma_1 \sqrt{\log(n)/n} \leq \mathrm{M}_{l_\infty^{(n)}} \leq \gamma_2 \sqrt{\log(n)/n}.$$

Proof of Lemma 3.3.12. Using Lemma 3.3.10, there is a constant $\lambda > 0$ and a vector $x \in S^{n-1}$ such that (i) $|l_\infty(x) - \mathrm{M}_{l_\infty}| \leq \lambda\sqrt{1/n}$, (ii) $|l(x) - \mathrm{M}_l| \leq \lambda \, \mathfrak{b}_l \, / \sqrt{n}$ and (iii) $|\{i : |x_i| > \frac{1}{K\sqrt{n}}\}| > \frac{n}{2}$ By Fact 3.3.13, $\mathrm{M}_{l_\infty} = \Theta(\sqrt{\log(n)/n})$. On the other hand, $\mathrm{mmc}(l) \leq \gamma\sqrt{n}$, for sufficiently small γ, thus $\lambda \, \mathfrak{b}_l \, / \sqrt{n} < \mathrm{M}_l$. We can therefore get constants $\gamma_1, \gamma_2 > 0$ such that $\gamma_1 \, \mathrm{M}_l \leq l(x) \leq \gamma_2 \, \mathrm{M}_l$ and $\gamma_1 \sqrt{\log(n)/n} \leq l_\infty(x) \leq \gamma_2 \sqrt{\log(n)/n}$. Therefore $|x| \leq \gamma_2 \sqrt{\log n} \xi^{(n)}$ coordinate-wise, and by monotonicity of symmetric norms,

$$\gamma_1 \, \mathrm{M}_l \leq l(x) \leq \gamma_2 \sqrt{\log n} \; l(\xi^{(n)}). \tag{3.3.2}$$

For the second part of the lemma, let $J = \{i : |x_i| > \frac{1}{K\sqrt{n}}\}$. As $|J| > \frac{n}{2}$, there is a permutation π such that $[n] - J \subset \pi(J)$. Let $|x|$ be a vector obtained from x by taking an absolute value of every coordinate, and let $\pi(x)$ denote applying permutation π to coordinates of vector x. We have $|x| + \pi(|x|) > \frac{\xi^{(n)}}{K}$ coordinate-wise, and therefore by monotonicity of symmetric norms, we have

$$\frac{1}{K} l(\xi^{(n)}) \leq l(|x| + \pi(|x|)) \leq l(|x|) + l(\pi(|x|)) = 2l(x) \leq 2\gamma_2 \, \mathrm{M}_l \, . \qquad \square$$

Next, we show that the median is roughly monotone (in n), which is crucial for the norm to be approximated.

Lemma 3.3.14 (Monotonicity of Median)**.** Let $l : \mathbb{R}^n \to \mathbb{R}$ be a symmetric norm. For all $n' \leq n'' \leq n$,

$$\mathrm{M}_{l^{(n')}} \leq \lambda \, \mathrm{mmc}(l) \sqrt{\log n'} \, \mathrm{M}_{l^{(n'')}},$$

where $\lambda > 0$ is an absolute constant.

Proof. By Lemma 3.3.12 and the fact that $\xi^{(n')}$ is also a vector in $S^{n''-1}$,

$$\lambda \, \mathrm{M}_{l^{(n')}} / \sqrt{\log n'} \leq l(\xi^{(n')}) \leq \mathfrak{b}_{l^{(n'')}} \leq \mathrm{mmc}(l) \, \mathrm{M}_{l^{(n'')}} \, . \qquad \square$$

We are now ready to prove the Lemma 3.3.8.

Proof of Lemma 3.3.8. Fix a β-contributing level i, and let U be the vector V after removing

buckets $j = 0, \ldots, i$. By Lemma 3.3.12, there is an absolute constant $\lambda_1 > 0$ such that

$$l(V_i) = \alpha^i \sqrt{b_i}\; l(\xi^{(b_i)}) \le \lambda_1 \alpha^i \sqrt{b_i}\, \mathrm{M}_{l^{(b_i)}},$$

and similarly

$$l(U) \ge \frac{\lambda_2 \alpha^i}{\sqrt{\log n}} \sqrt{\sum_{j>i} b_j}\;\; \mathrm{M}_{l^{(\sum_{j>i} b_j)}}\,.$$

We now relate these two inequalities as follows. First, $l(V_i) \ge \beta\, l(V) \ge \beta\, l(U)$. Second, we may assume $b_i < \sum_{j>i} b_j$, as otherwise the lemma holds, and then by monotonicity of the median (Lemma 3.3.14) $\mathrm{M}_{l^{(b_i)}} \le \lambda_3 \,\mathrm{mmc}(l) \sqrt{\log n}\; \mathrm{M}_{l^{\sum_{j>i} b_j}}$, for some absolute constant $\lambda_3 > 0$. Putting these together, we get

$$\beta \cdot \frac{\lambda_2 \alpha^i}{\sqrt{\log n}} \sqrt{\sum_{j>i} b_j} \le \lambda_1 \alpha^i \sqrt{b_i} \cdot \lambda_3 \,\mathrm{mmc}(l) \sqrt{\log n},$$

and the lemma follows. □

Lemma 3.3.15. If level i is β-contributing, then there is an absolute constant $\lambda > 0$ such that

$$b_i \alpha^{2i} \ge \frac{\lambda \beta^2}{\mathrm{mmc}(l)^2 (\log_\alpha n) \log^2 n} \sum_{j \le i} b_j \alpha^{2j}.$$

Proof of Lemma 3.3.15. Fix a β-contributing level i, and let $h := \mathrm{argmax}_{j \le i} \sqrt{b_j} \alpha^j$. We

proceed by separating into two cases. First, if $b_i \geq b_h$ then the lemma follows easily by

$$\sum_{j \leq i} b_j \alpha^{2j} \leq t b_h \alpha^{2h} \leq O(\log_\alpha n) b_i \alpha^{2i}.$$

The second case is when $b_i < b_h$. Using the definition of a contributing level and Lemma 3.3.12,

$$\lambda_1 \alpha^i \sqrt{b_i}\, \mathrm{M}_{l^{(b_i)}} \geq l(V_i) \geq \beta l(V) \geq \lambda_2 \beta \alpha^h \sqrt{b_h / \log n}\, \mathrm{M}_{l^{(b_h)}},$$

for some absolute constants $\lambda_1, \lambda_2 > 0$. Plugging in $\mathrm{M}_{l^{(b_i)}} \leq \lambda_3 \operatorname{mmc}(l) \sqrt{\log n}\, \mathrm{M}_{l^{(b_h)}}$, which follows from monotonicity of the median (Lemma 3.3.14), for some absolute constant $\lambda_3 > 0$, we get

$$\begin{aligned} \lambda_1 \alpha^i \sqrt{b_i}\, \mathrm{M}_{l^{(b_i)}} &\geq \frac{\lambda_2 \beta \sqrt{b_h} \alpha^h}{\sqrt{\log n}} \cdot \frac{\mathrm{M}_{l^{(b_i)}}}{\lambda_3 \operatorname{mmc}(l) \sqrt{\log n}}, \\ \sqrt{b_i} \alpha^i &\geq \frac{\lambda_2 \beta \sqrt{b_h} \alpha^h}{\lambda_1 \lambda_3 \operatorname{mmc}(l) \log n}. \end{aligned}$$

Squaring the above and observing that $b_h \alpha^{2h} \geq \frac{1}{O(\log_\alpha n)} \sum_{j \leq i} b_j \alpha^{2j}$, the proof is complete.

□

3.3.4 Putting It Together

Proof of Theorem 3.1.1. Recall from Section 3.2 that we assume our algorithm has access to an oracle `NORM` that computes $l(v)$ using queries to the coordinates of v, i.e., our algorithm must provide query access to any coordinate v_i. We assume without loss of generality that $\epsilon \geq 1/\mathrm{poly}(n)$, because an exact algorithm using space $O(n \log n)$ is trivial.

Our algorithm maintains a data structure that eventually produces a vector $\widehat{V}$. We will show that with high probability, $l(\widehat{V})$ approximates $l(v)$ and we will also bound the space required for the data structure. The algorithm is presented in Algorithm 3. The idea is to run the `Level1` algorithm with appropriate parameters. Specifically, to achieve $(1 \pm \epsilon)$-approximation to $l(v)$, we set the approximation guarantee of the buckets to be $\epsilon' := O\left(\frac{\epsilon^2}{\log n}\right)$ and the importance guarantee to be $\beta' := O\left(\frac{\epsilon^5}{\mathrm{mmc}(l)^2 \log^5 n}\right)$.

Algorithm 3 `OnePassSymmetricNorm`$(\mathcal{S}, n)$

1: **Input:** stream $\mathcal{S}$ of from domain $[n]$, and $\epsilon > 0$
2: **Output:** X
3: $(\alpha, \widehat{b}_1, \widehat{b}_2, \ldots, \widehat{b}_t) \leftarrow$ `Level1`$(\mathcal{S}, n, \alpha' = 1 + O(\epsilon), \epsilon' = O\left(\frac{\epsilon^2}{\log n}\right), \beta' = O\left(\frac{\epsilon^5}{\mathrm{mmc}(l)^2 \log^5 n}\right), \delta = \frac{0.01\epsilon}{n})$;
4: Construct $\widehat{V}$ using α and $\widehat{b}_1, \widehat{b}_2, \ldots, \widehat{b}_t$;
5: Invoke `NORM`, answer each query for v_i by $\widehat{V}_i$;
6: $X \leftarrow$ output of `NORM`.
7: Return X.

Let v be the streaming vector. It is approximated by its level vector V with base $\alpha = 1 + O(\epsilon)$, namely, $(1 - O(\epsilon))l(v) \le l(V) \le l(v)$ by Proposition 3.3.4. Observe that $t = O(\log_\alpha n) = O(\log(n)/\epsilon)$, and assume that algorithm `Level1` succeeds, i.e., the high-probability event in Theorem 3.3.2 indeed occured. Denote by $\widehat{V}$ the output of `Level1`, and by V' the vector V after removing all buckets that are not β-contributing, and define $\widehat{V}'$ similarly to $\widehat{V}$, where we set $\beta := \epsilon/t = O(\epsilon^2/\log n)$. Every β-contributing level is necessarily β'-important by Lemmas 3.3.8 and 3.3.15 and therefore satisfies $\widehat{b}_i \ge (1 - \epsilon')b_i$. We bound the error from removing non-contributing levels by Lemma 3.3.7, namely,

$$(1 - O(\epsilon))\ l(V) \le (1 - O(\log_\alpha n) \cdot \beta)\ l(V) \le l(V') \le l(V).$$

By monotonicity (Lemma 3.2.1) and by Lemma 3.3.5,

$$\begin{aligned} l(\widehat{V}) \geq l(\widehat{V}') &= l((V' \backslash V_{i_1} \cup \widehat{V}_{i_1}) \dots \backslash V_{i_k} \cup \widehat{V}_{i_k}) \\ &\geq (1-\epsilon')^t \; l(V') \geq (1 - O(\epsilon)) l(V'). \end{aligned}$$

Altogether, $(1 - O(\epsilon))l(v) \leq l(\widehat{V}') \leq l(v)$, which bounds the error of $l(\widehat{V}')$ as required.

The space requirement of the algorithm is dominated by that of subroutine `Level1`, namely, $O\left(\frac{\log^{12} n}{\beta' \epsilon'^2 \epsilon^5}\right) = O\left(\frac{\mathrm{mmc}(l)^2 \log^{19} n}{\epsilon^{14}}\right)$ bits. Storing the data structure, i.e., $\widehat{b}_i$'s, requires only

$O(\log_\alpha n) \log n = O\left(\frac{\log^2 n}{\epsilon}\right)$ bits. □

3.4 Lower Bound

The overall plan is to use the multiparty disjointness communication complexity problem to prove an $\Omega(\mathrm{mmc}(l)^2)$ bits storage lower bound on any turnstile streaming algorithm outputs a $(1 \pm 1/6)$-approximation, or better, to the norm of the frequency vector. The bound is otherwise independent of the norm or n.

Multiparty disjointness is a communication problem where there are t players who each recieve a subset of $[n]$, and their goal is to determine whether their sets are intersecting or not. The problem was introducted by Alon, Matias, and Szegedy[14] to prove storage lower bounds for the frequency moments problem. After several improvements,[17,70] the communication complexity of multiparty disjointness was settled at an asymptotically optimal

$\Omega(n/t)$ bits of communication by Gronemeier.[71]

3.4.1 John's Theorem for Symmetric Norms

We will start by proving the following specialization of John's Theorem[96] to the case of symmetric norms.

Theorem 3.4.1 (John's Theorem for Symmetric Norms)**.** If $l(\cdot)$ is a symmetric norm on $\mathbb{R}^n$, then there exist $0 < a \leq b$ such that $b/a \leq \sqrt{n}$ and, for all $x \in \mathbb{R}^n$, $al_2(x) \leq l(x) \leq bl_2(x)$.

Proof. By John's Theorem[96] there exists a unique ellipsoid E of maximum volume contained in $B = \{x \in \mathbb{R}^n \mid l(x) \leq 1\}$ and, furthermore, $B \subseteq \sqrt{n}E$. E is permutation and sign symmetric because B is, so it follows from Lemma 3.4.2 that E is a sphere. Therefore, there exist $0 < a < b$ such that $al_2(x) \leq l(x) \leq bl_2(x)$, for all $x \in \mathbb{R}^n$, and, furthermore, $b/a \leq \sqrt{n}$. □

Lemma 3.4.2. If an ellipsoid E is symmetric under every permutation or change of signs to its coordinates then E is a sphere.

Proof. Let A be a positive semidefinite matrix such that $E = \{x \in \mathbb{R}^n \mid x^T Ax = 1\}$. Since A is a real positive semidefinite matrix, it can be decomposed as $A = SDS^T$, where S is orthonormal and D is a diagonal matrix with $D_{11} \geq D_{22} \geq \cdots D_{nn} \geq 0$. We will show that all of the diagonal entries in D are the same, from which it follows that $A = D$ and E is a sphere. Let s_i, for $i \in [n]$, be the columns of S. Let $i \neq 1$, choose a permutation P_1 so that $P_1 s_1$ has its coordinates in decreasing order by magnitude, and choose a permutation P_i so that $P_i s_i$ has the same. Now choose a diagonal matrix D that has $D_{jj} = 1$ if $(P_1 s_1)_j$

has the same sign as $(P_i s_i)_j$, and $D_{jj} = -1$ if the signs are different, zeros may be treated arbitrarily. Let $P = P_1^T D P_i$; since P is the product of permutation matrics and a sign change matrix we have $E = \{x \mid x^T P A P^T x \leq 1\}$ by the symmetry assumption.

We have $D_{11} = s_1^T A s_1$, since s_1 is a unit vector orthogonal to s_i, $i > 1$. Let $\lambda = S^T P^T s_1$. By construction we have $\sum_j \lambda_j^2 = 1$ and $\lambda_i > 0$. If we suppose that $D_{ii} < D_{11}$, then we arrive at the following contradiction

$$D_{11} = s_1 A s_1 = s_1 P A P^T s_1 = \sum_{j=1}^{n} \lambda_j^2 D_{jj} < D_{11}.$$

Therefore, $D_{11} = D_{ii}$, for all i, and E is sphere. □

3.4.2 Concentration of a Symmetric Norm

Let us begin by discussing a concentration inequality for symmetric norms. We will need concentration of $l(Z)$ around $\sqrt{n}\,\mathrm{M}_l$, where Z is distributed according to the canonical Gaussian disribution on n dimensions. To get it, we will use the following two concentration theorems for Lipschitz functions. The diffrence between them is the underlying distribution, whether it is uniform on S^{n-1} or multivariate Gaussian. Comparing $l(Z)$ against its own median is just a direct application of Theorem 3.4.4. There is a little bit more work to do because we wish to compare $l(Z)$ to the median of $l(\cdot)$ over S^{n-1}, which is also the median of $l(Z)/l_2(Z)$. Note that the M in Theorem 3.4.3 is not the same as the M in Theorem 3.4.4 because the probability distributions are different.

Theorem 3.4.3 ([82])**.** Let $f : S^{n-1} \to \mathbb{R}$ be 1-Lipschitz, let $Z \in S^{n-1}$ be chosen uniformly

at random, and let M be the median of $f(Z)$. Then, for all $t > 0$, $\Pr(|f(Z) - M| \geq t) \leq 2e^{-nt^2/2}$.

Theorem 3.4.4 ([97])**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be 1-Lipschitz, let $Z_1, Z_2, \ldots, Z_n \overset{\text{iid}}{\sim} N(0,1)$, and let M be the median of $f(Z)$. Then, for all $t > 0$, $\Pr(|f(Z) - M| \geq t) \leq e^{-t^2/2}$.

It will also be helpful to have the following fact about χ^2 random variables.

Lemma 3.4.5. ([98]) Let $X \sim \chi^2_n$. For all $x \geq 0$,

$$\Pr(X \geq n + 2\sqrt{nx} + x) \leq e^{-x} \quad \text{and} \quad \Pr(X \leq n - 2\sqrt{nx}) \leq e^{-x}.$$

Lemma 3.4.6. Let $n \geq 2$ and let $Z \in \mathbb{R}^n$ be a random vector with coordinates $Z_1, Z_2, \ldots, Z_n \overset{\text{iid}}{\sim} N(0,1)$. Let M_l be the median of $l(\cdot)$ on S^{n-1}, where $l(\cdot)$ is a symmetric norm on $\mathbb{R}^n$. Then, for all $t \geq 0$,

$$\Pr(|l(Z) - \sqrt{n}\,\mathrm{M}_l\,| \geq t\sqrt{n}\,\mathrm{M}_l) \leq 7e^{-t^2/200}.$$

Proof. We first establish an inequality that does not have the correct dependence on t, it is (3.4.2), and then use it to bound the median of $l(Z)$ in terms of $\sqrt{n}\,\mathrm{M}_l$. That will allow us to apply Theorem 3.4.4 and get the bound above.

By Theorem 3.4.1, there exist $0 < a_l \leq b_l$ such that $b_l/a_l \leq \sqrt{n}$ and, for all $x \in \mathbb{R}^n$, $a_l l_2(x) \leq l(x) \leq b_l l_2(x)$. This implies $l(\cdot)$ is b_l-Lipschitz on $\mathbb{R}^n$. By scaling the norm (and, as a consequence, M_l), we may assume $a_l = 1$ without loss of generality.

It is easy to see that

$$\begin{aligned}
&\Pr(l(Z) - \sqrt{n}\,\mathrm{M}_l \geq t\sqrt{n}\,\mathrm{M}_l) \\
&= \Pr\left(l(Z) - l_2(Z)\,\mathrm{M}_l + l_2(Z)\,\mathrm{M}_l - \sqrt{n}\,\mathrm{M}_l \geq t\sqrt{n}\,\mathrm{M}_l\right) \\
&\leq \Pr\left(l(Z) - l_2(Z)\,\mathrm{M}_l \geq \sqrt{n}\,\mathrm{M}_l\,\frac{t}{2}\right) + \Pr\left(l_2(Z) - \sqrt{n} \geq \sqrt{n}\frac{t}{2}\right). \qquad (3.4.1)
\end{aligned}$$

For the second term, notice that $l_2(Z)^2$ is a χ^2_n random variable. Using Lemma 3.4.5, we have

$$\begin{aligned}
\Pr\left(l_2(Z) - \sqrt{n} \geq \sqrt{n}\frac{t}{2}\right) &= \Pr\left(l_2(Z)^2 \geq n(1+\frac{t}{2})^2\right) \\
&= \Pr\left(l_2(Z)^2 \geq n + 2\sqrt{n}(\sqrt{n}\frac{t}{2}) + (\sqrt{n}\frac{t}{2})^2\right) \leq e^{-nt^2/4}.
\end{aligned}$$

For the first term in (3.4.1), we have

$$\begin{aligned}
&\Pr\left(|l(Z) - l_2(Z)\,\mathrm{M}_l| \geq \sqrt{n}\,\mathrm{M}_l\,\frac{t}{2}\right) \\
&\leq \Pr\left(\left|l(\frac{Z}{l_2(Z)}) - \mathrm{M}_l\right| \geq \mathrm{M}_l\,\frac{t}{4}\right) + \Pr(l_2(Z) \geq 2\sqrt{n}).
\end{aligned}$$

The scaled norm $l(\cdot)/\,\mathfrak{b}_l$ is 1-Lipschitz and $Z/l_2(Z)$ is distributed according to the Haar distribution, so by Theorem 3.4.3 and our previous χ^2 bound we have

$$\begin{aligned}
\Pr\left(l(Z) - l_2(Z)\,\mathrm{M}_l \geq \sqrt{n}\,\mathrm{M}_l\,\frac{t}{2}\right) &\leq 2\exp\{-\frac{n\,\mathrm{M}_l^2\,t^2}{32\,\mathfrak{b}_l^2}\} + e^{-n} \\
&\leq 2\exp\{-\frac{t^2}{32}\} + e^{-n}
\end{aligned}$$

where the final inequality follows because $\mathrm{M}_l \,/\, \mathfrak{b}_l \geq a_l/b_l \geq 1/\sqrt{n}$.

So far, we have established, $\forall\, t \geq 0$,

$$\Pr(l(Z) - \sqrt{n}\,\mathrm{M}_l \geq t\sqrt{n}\,\mathrm{M}_l) \leq 2e^{-t^2/32} + e^{-n} + e^{-nt^2/4}. \tag{3.4.2}$$

It is almost the bound that we want, except for the e^{-n} term. Substituting in $t = 8$ and $n \geq 2$ we find $\Pr(l(Z) \geq 9\sqrt{n}\,\mathrm{M}_l) \leq \frac{1}{2}$. Therefore the median of $l(Z)$ is at no larger than $9\sqrt{n}\,\mathrm{M}_l$, so Theorem 3.4.4 implies, $\forall\, t \geq 0$ and $n \geq 2$,

$$\Pr(l(Z) - 9\sqrt{n}\,\mathrm{M}_l \geq t\sqrt{n}\,\mathrm{M}_l) \leq e^{-t^2 n\,\mathrm{M}_l^2\,/2\,\mathfrak{b}_l^2} \leq e^{-t^2/2}. \tag{3.4.3}$$

The last step is to combine these two bounds by using (3.4.2) to bound,

$$\forall t \leq 10 \text{ and } n \geq 2,\ \ \Pr(l(Z) \geq t\sqrt{n}\,\mathrm{M}_l) \leq 3e^{-t^2/32} + e^{-n} \leq 7e^{-t^2/32}$$

and using (3.4.3) to establish, $\forall t \geq 10$ and $n \geq 2$,

$$\begin{aligned}\Pr(l(Z) \geq t\sqrt{n}\,\mathrm{M}_l) &= \Pr(l(Z) - 9\sqrt{n}\,\mathrm{M}_l \geq (t-9)\sqrt{n}\,\mathrm{M}_l) \\ &\leq \Pr(l(Z) - 9\sqrt{n}\,\mathrm{M}_l \geq \frac{t}{10}\sqrt{n}\,\mathrm{M}_l) \leq e^{-t^2/200},\end{aligned}$$

which proves the theorem. □

3.4.3 The Norm of a Randomized Vector

The multiparty disjointness reduction used to prove Theorem 3.1.2 uses a randomized vector. Given a vector $v \in \mathbb{R}^n$, we randomize it by replacing the coordinates by independent Normally distributed random variables $V_i \sim N(0, v_i^2)$, for each $i \in [n]$.

The next lemma allows us to compare the distribution of the norm of two different randomized vectors. Recall that a random variable Y is said to *stochastically dominate* a random variable X if $\Pr(Y \geq t) \geq \Pr(X \geq t)$ for all $t \in \mathbb{R}$, or, equivalently, their cdf's satisfy $F_X \geq F_Y$.

Lemma 3.4.7. Let $\sigma, \tau \in \mathbb{R}^n_{\geq 0}$ satisfy $\sigma \leq \tau$ coordinate-wise. Let $X_i \sim N(0, \sigma_i^2)$, independently for $i = 1, \ldots, n$, and $Y_i \sim N(0, \tau_i^2)$, independently for $i = 1, \ldots, n$. Then $l(Y)$ stochastically dominates $l(X)$, in particular, for all $t \in \mathbb{R}$,

$$\Pr(l(X) \geq t) \leq \Pr(l(Y) \geq t).$$

Proof. It is well known that, for any random variables Y' and X', Y' stochastically dominates X' if and only if there is a coupling of X' and Y' so that $X' \leq Y'$. Since $\tau_i \geq \sigma_i$ we have that $|Y_i|$ stochastically dominates $|X_i|$, for all i. Therefore, there is a coupling of the vectors $|X|$ and $|Y|$ so that $|X| \leq |Y|$ coordinate-wise at every sample point. This is also a coupling of $l(X)$ and $l(Y)$, and by applying Lemma 3.2.1 proves that $l(X) \leq l(Y)$ at every sample point. Thus, $l(Y)$ stochastically dominates $l(X)$. □

The main technical lemma we use to prove Theorem 3.1.2 is the following.

Lemma 3.4.8. If $v \in S^{n-1}$ has $l(v) = \mathfrak{b}_l$ and $V \in \mathbb{R}^n$ is a random vector with coordinates distributed $V_i \overset{\text{iid}}{\sim} N(0, v_i^2)$, then $\Pr(l(V) \geq \mathfrak{b}_l/4) \geq 1/10$.

In order to prove Lemma 3.4.8 we will first need to bound $\mathbb{E}\, l(V)$.

Lemma 3.4.9. If $v \in S^{n-1}$ has $l(v) = \mathfrak{b}_l$ and V is a random vector with coordinates distributed $V_i \overset{\text{iid}}{\sim} N(0, v_i^2)$, then $\mathbb{E}\, l(V) \geq 0.49\, \mathfrak{b}_l$.

Proof. If on every outcome it happened that $|V| \geq |v|$ coordinate-wise then Lemma 3.2.1 would imply the desired result. Of course, it is very likely that for some coordinates $|V_i| < |v_i|$. The idea of the proof is to "patch up" those coordinates with another vector that has small norm and then apply the reverse triangle inequality. Let $U = \max\{|v| - |V|, 0\}$, where the maximum is taken coordinate-wise. U was chosen so that $|V| + U \geq |v|$, hence by Lemma 3.2.1 $l(|V|+U) \geq l(v) = \mathfrak{b}_l$, and by the reverse triangle inequality $l(V) \geq l(v) - l(U)$.

It remains to bound $\mathbb{E}\, l(U)$. We will begin by bounding $\mathbb{E}\, l_2(U)$ and use this value to bound $\mathbb{E}\, l(U)$. Let $Z \sim N(0,1)$ and let $\mathbf{1}_A$ be the indicator function of the set A. Direct calculation with the Normal c.d.f. shows that

$$\mathbb{E}\, l_2(U)^2 = \sum_{i=1}^{n} 2v_i^2\, \mathbb{E}\left(\mathbf{1}_{[0,1)}(Z)\,(1-Z)^2\right) \leq 0.26 \sum v_i^2 = 0.26,$$

Therefore, $\mathbb{E}\, l_2(U) \leq \left(\mathbb{E}\, l_2(U)^2\right)^{1/2} \leq 0.51$, where the first inequality is Jensen's. Finally, we can conclude $\mathbb{E}\, l(U) \leq \mathfrak{b}_l\, \mathbb{E}\, l_2(U) \leq 0.51\, \mathfrak{b}_l$ and $\mathbb{E}\, l(V) \geq l(v) - \mathbb{E}\, l(U) \geq 0.49\, \mathfrak{b}_l$. □

Proof of Lemma 3.4.8. For a random variable X and event A, let $\mathbb{E}(X; A) = \mathbb{E}\, X\mathbf{1}_A(X) =$

$\int_A X dP$. We begin with the trivial bound, for any $0 < \alpha < \beta$,

$$\begin{aligned}\mathbb{E}\, l(V) &= \mathbb{E}(l(V); (0, \alpha]) + \mathbb{E}(l(V); (\alpha, \beta]) + \mathbb{E}(l(V); (\beta, \infty)) \\ &\leq \alpha + \beta \Pr(l(V) \in (\alpha, \beta]) + \mathbb{E}(l(V); (\beta, \infty)). \end{aligned} \tag{3.4.4}$$

We shall use $l_2(V)$ to bound the last term above. Observe that $\mathbb{E}\, l_2(V)^2 = 1$ and, letting $Z_1, \ldots, Z_n \overset{\text{iid}}{\sim} N(0,1)$,

$$\begin{aligned}\operatorname{Var}(l_2(V)^2) &= \operatorname{Var}\left(\sum_i v_i^2 Z_i^2\right) \\ &= \sum_i v_i^4 \operatorname{Var}(Z_i^2) = 2\sum_i v_i^4 \leq 2 l_2(v)^2 = 2,\end{aligned}$$

because $v \in S^{n-1}$ has unit length. For $k > 0$, we have by Chebyshev's Inequality that

$$\Pr(l_2(V)^2 \geq \sqrt{2}k + 1) \leq \frac{1}{k^2},$$

and, by a change of variables,

$$\Pr(l_2(V) \geq x) \leq \left(\frac{x^2 - 1}{\sqrt{2}}\right)^{-2} = \frac{2}{(x^2-1)^2} \leq 4/x^4, \qquad \text{for } x > \sqrt{2},$$

and it extends trivially to all $x > 0$. This implies $\Pr(l(V) \geq \mathfrak{b}_l\, x) \leq 4/x^4$, hence $\Pr(l(V) \geq x) \leq 4(\mathfrak{b}_l\, /x)^4$. Thus,

$$\mathbb{E}(l(V); (\beta, \infty)) \leq \int_\beta^\infty \frac{4\,\mathfrak{b}_l^4}{x^4} dx = \frac{4\,\mathfrak{b}_l^4}{3\beta^3}.$$

Now we return to (3.4.4) and substitute $\alpha = 0.49\,\mathfrak{b}_l\, /4$ and $\beta = 2.44\,\mathfrak{b}_l$. Together with

Lemma 3.4.9 we get

$$0.49\,\mathfrak{b}_l \le \mathbb{E}\, l(V) \le \frac{0.49\,\mathfrak{b}_l}{4} + 2.32\,\mathfrak{b}_l \Pr(l(V) \ge \mathfrak{b}_l\,/4) + \frac{4\,\mathfrak{b}_l}{3(2.44)^3}.$$

Upon rearranging the inequality we find $\Pr(l(V) \ge \mathfrak{b}_l\,/4) \ge 1/10$, as desired. □

3.4.4 Multiparty Disjointness and the Norm on a Stream

We will show an $\Omega(\mathrm{mc}(l)^2)$ bits bound on the storage needed by a streaming algorithm for the norm l.

Lemma 3.4.10. Let $l(\cdot)$ be a symmetric norm on $\mathbb{R}^n$. A turnstile streaming algorithm that outputs a $(1 \pm \frac{1}{6})$-approximation for $l(\cdot)$, with probability at least 0.99, uses $\Omega(\mathrm{mc}(l)^2)$ bits in the worst case.

Let recall that every symmetric norm $l(\cdot)$ on $\mathbb{R}^n$ induces the norm $l(\cdot)^{(k)}$ on $\mathbb{R}^k$, for $k < n$, by setting any $n - k$ coordinates to 0. The induced norm may have a different ratio of $\mathfrak{b}_l\,/\,\mathrm{M}_l$. Since a streaming algorithm that approximates $l(\cdot)$ must also approximate $l(\cdot)^{(k)}$, Lemma 3.4.10 in fact implies Theorem 3.1.2.

Proof of Lemma 3.4.10. We begin with an instance of the multiparty disjointness promise problem on domain $[n]$ with $t = \lceil 240\sqrt{n} \cdot \mathrm{M}_l\,/\,\mathfrak{b}_l \rceil$ players. By Lemma 3.4.1, $t \ge 240\sqrt{n}\,\mathrm{M}_l\,/\,\mathfrak{b}_l \ge 240$. The players are given sets $P_1, P_2, \ldots, P_t \subseteq [n]$ with the promise that either they are pairwise disjoint or exactly one element is contained in every set but they are otherwise disjoint. The players are allowed, in any order, to communicate bits with each other by writing

them to a shared blackboard, and they are given shared access to a string of random bits. The players' goal is for at least one among them to determine whether the sets $P_1, \ldots, P_t$ are disjoint or intersecting. If the players correctly determine the type of instance with probability at least 0.55, then their communication scheme is called a "correct protocol". It is known that for any correct protocol, the players must write $\Omega(n/t)$ bits to the blackboard in the worst case.[70] In this reduction, each of the t players will transmit the memory of the streaming algorithm once, which leads to an $\Omega(n/t^2) = \Omega(\mathfrak{b}_l^2 \,/\, \mathrm{M}_l^2)$ bits lower bound on the memory used by the algorithm.

Next, we describe the protocol under the assumption that the players can perform computations with real numbers. After describing the protocol we explain that this assumption can be removed by rounding the real values to a sufficiently high precision.

The players have shared access n^2 i.i.d. $N(0,1)$ random variables $Z_{i,j}$, for $i, j \in [n]$, and additional independent randomness for the approximation algorithm.

Let $v \in \operatorname{argmax}_{x \in S^{n-1}} l(x)$, so that $l(v) = \mathfrak{b}_l$. We define an $n \times n$ matrix V with coordinates

$$V_{i,j} = Z_{i,j} v_{i+j \bmod n}.$$

Since v is fixed, all of the players can compute the matrix using the shared randomness. Let V_j denote the jth column of V; it is a vector with independent Normally distributed entries. The vector of standard deviations of V_j is a copy of v that has been cyclically shifted down by j entries, in particular the standard deviation of $V_{i,n}$ is v_i.

Here is the stream that the players create, they jointly run an approximation algorithm

for the norm on this stream. For each player i and item $j \in P_i$ the ith player adds a copy of V_j to the stream. More precisely, for each $j \in P_i$ player i adds 1 with frequency $V_{1,j}$, 2 with frequency $V_{2,j}$, etc. The players repeat this protocol 10 times independently.

Now we analyze the possible outcomes of one of the ten trials. Let $N \subseteq \cup_{i=1}^{t} P_i$ be the set of elements that appear in exactly one set P_i, and let $X = \sum_{j \in N} V_j$. If there is no intersection between the P_i's, then the stream's frequency vector is X. If they all intersect at j^*, then the frequency vector is $Y = tV_{j^*} + X$.

It remains to compare $l(X)$ and $l(Y)$. The coordinates of X are independent and normally distributed with zero mean and variance

$$\mathbb{E}\, X_i^2 = \sum_{j \in N} \mathbb{E}\, V_{i,j}^2 = \sum_{j \in N} v_{i+j \bmod n}^2 \leq \sum_{j=1}^{n} v_j^2 = 1.$$

Let Z be a random vector with coordinates $Z_i \overset{\text{iid}}{\sim} N(0,1)$, for $i = 1, \ldots, n$. By Lemma 3.4.7 Z stochastically dominates X, and using Lemma 3.4.6 we have $\Pr(\frac{1}{\sqrt{n}} l(X) \geq 40\,\mathrm{M}_l) \leq \Pr(\frac{1}{\sqrt{n}} l(Z) \geq 40\,\mathrm{M}_l) \leq 0.005$. On the other hand, Y stochastically dominates tV_{j^*} and Lemma 3.4.8 additionally implies

$$\Pr(l(tV_{j^*} + X) \geq 60\,\mathrm{M}_l \sqrt{n}) \geq \Pr(tl(V_{j^*}) \geq 60\,\mathrm{M}_l \sqrt{n}) \geq \Pr(l(V_{j^*}) \geq \mathfrak{b}_l\,/4) \geq 1/10.$$

The final player checks whether the maximum approximation returned among the 10 trials is larger or smaller than $50\,\mathrm{M}_l \sqrt{n}$ and declares "intersecting" or "disjoint" accordingly.

The output of the protocol is correct on an intersecting instance if at least one of the 10

stream vectors has norm larger than $60\,\mathrm{M}_l\sqrt{n}$ and the algorithm always returns a $(1\pm 1/6)$-approximation. It is correct on a disjoint instance if all of the stream vectors have norm smaller than $40\,\mathrm{M}_l\sqrt{n}$ and the algorithm always returns a $(1\pm 1/6)$-approximation. If the instance is an intersecting one, then with probability at least 0.1 the magnitude of $l(tV_{j^*}+X)$ is large enough. At least one of the ten trials will have this property with probability at least $1-0.9^{10}\geq 0.65$, because the trials use independent random matrices. Since the algorithm correctly approximates the norm with probability at least 0.99, it follows that the protocol is correct for an intersecting instance with probability at least $0.65-10\cdot 0.01=0.55$.

On a disjoint instance, one trial of the protocol is successful with probability at least $0.99^2\geq 0.98$ where one factor comes from the success of the approximation algorithm and the other from our earlier application of the concentration bound. Thus, the output of the protocol correctly identifies a disjoint instance with probability at least $1-10\cdot 0.02=0.8$, by a union bound over the ten trials. Therefore, this protocol is a correct protocol.

It remains to describe the rounding of the real values. It suffices to represent each value with a sufficiently high precision. We replace each variable as $Z_{i,j}$ with a discrete random variable $Z_{i,j}=\widehat{Z}_{i,j}+\delta_{i,j}$ where $\widehat{Z}_{i,j}$ are distributed i.i.d. $N(0,1)$ and $\delta_{i,j}$ is difference between $\widehat{Z}_{i,j}$ and its closest point in $\{\frac{j}{n^4}\mid j=-n^5,\ldots,n^5-1,n^5\}$. In particular, with very high probability, $|\delta_{i,j}|\leq 1/2n^4$ for all pairs i,j. We also replace v by a vector $v=\widehat{v}+\delta_v$ where $\widehat{v}\in\operatorname{argmax}_{x\in S^{n-1}} l(x)$, so that $l(\widehat{v})=\mathfrak{b}_l$, and where δ_v is a vector containing the difference between each entry of $\widehat{v_i}$ and the nearest integer multiple of n^{-4} to it.

Each frequency in the stream is the sum of at most t variables. Performing these

replacements changes each frequency in the stream by no more than $2tn^{-4}$. Let $\Delta \in \mathbb{R}^n$ denote this change, then $l(\Delta) \leq \mathfrak{b}_l \, l_2(\Delta) \leq 2 \, \mathfrak{b}_l \, tn^{-7/2} = O(\mathrm{M}_l / n^3)$. Applying the triangle and reverse triangle inequalities shows that the change negligible. Therefore, the discretized protocol is correct also, which completes the proof. □

Suppose there is an algorithm with the weaker, D-approximation guarantee. Namely, $D > 1$ and with probability at least 0.99, the algorithm returns a value $\widehat{l}$ satisfying $l(V) \leq \widehat{l} \leq Dl(V)$, where V is the stream vector. The main lower bound, Theorem 3.1.2, can be easily adapted this setting, where we get a lower bound of $\Omega(\mathrm{mmc}(l)^2/D^2)$ bits instead, with a small modification to the proof of Lemma 3.4.10.

Theorem 3.1.4 . Let l be a symmetric norm on $\mathbb{R}^n$. Any turnstile streaming algorithm that outputs, with probability at least 0.99, a D-approximation for $l(\cdot)$ must use $\Omega(\mathrm{mmc}(l)^2/D^2)$ bits of space in the worst case.

Indeed, the proof goes as above, except that the number of players should be increased to $t = \lceil 240D\sqrt{n}\,\mathrm{M}_l \,/\, \mathfrak{b}_l \rceil$. The disjoint instances do not change, but the norm is D times larger on an intersecting instance. Thus, the D-approximation algorithm can distinguish the two and we get the bound $\Omega(\mathrm{mc}(l)^2/D^2)$, which is easily boosted to $\Omega(\mathrm{mmc}(l)^2/D^2)$ bits, as before. When $l = l_\infty$, this matches the trade-off proved by Saks and Sun.[94]

3.5 Optimal Space-Approximation Tradeoff

In this section we obtain a nearly tight space-approximation tradeoff for computing any symmetric norm in the data-stream model. Specifically, we show below how our earlier

streaming algorithm can be adapted to match the lower bound in Theorem 3.1.4, up to a polylog(n) factor. The adapted algorithm achieves, for any $D \geq 1.1$ and symmetric norm l, a D-approximation within $\widetilde{O}(\mathrm{mmc}(l)^2/D^2)$ bits of storage. The key part of the analysis is to define, a new symmetric function $l_{(D)}$ on $\mathbb{R}^n$ such that $l(x) \leq l_{(D)}(x) \leq Dl(x)$, for all $x \in \mathbb{R}^n$, and such that our earlier algorithm can find a $(1 \pm 1/2)$-approximation to $l'(x)$ using $\mathrm{polylog}(n) \cdot \mathrm{mmc}(l)^2/D^2$ bits of space.

We start in Section 3.5.1 with an algorithm for Q-norms (formaly defined in Section 3.6.2), a special case that is easier to prove. We then leverage ideas from this simpler case to design in Section 3.5.2 an algorithm for general symmetric norm.

3.5.1 D-Approximation for Q-norms

Theorem 3.5.1. Let $l : \mathbb{R}^N \to \mathbb{R}$ be a Q-norm. Then for every $1.1 < D \leq \mathrm{mmc}(l)$ there is a randomized streaming algorithm that D-approximates l and uses $\widetilde{O}(\mathrm{mmc}(l)^2/D^2)$ bits of space.

Proof. Fix a Q-norm l and $1 < D \leq \mathrm{mmc}(l)$. We first show that for all $x \in \mathbb{R}^N$,

$$l_{(D)}(x) := \max\left(\frac{D\,\mathrm{M}_l\, l_2(x)}{\log n}, l(x)\right)$$

is an $O(D)$-approximation to $l(x)$. Since l is a Q-norm, we have by Lemma 3.6.9 that $\xi^{(n)}$ is roughly a minimizer of $l(x)$ over S^{N-1}, namely,

$$\forall x \in \mathbb{R}^{N-1}, \qquad l(\xi^{(n)})\; l_2(x) \leq 6\sqrt{\log n}\; l(x).$$

Recalling from Lemma 3.3.12 that, for some absolute constants $\lambda_1, \lambda_2 > 0$, $\lambda_1 \mathrm{M}_l / \sqrt{\log n} \leq l(\xi^{(n)}) \leq \lambda_2 \mathrm{M}_l$, we have that $\lambda_1 \mathrm{M}_l \, l_2(x) \leq 6(\log n) \cdot l(x)$. Altogether we obtain (assuming, without loss of generality, that $\lambda_1 < 1$)

$$\forall x \in \mathbb{R}^{N-1}, \qquad l(x) \leq l_{(D)}(x) \leq \frac{6D}{\lambda_1} l(x). \tag{3.5.1}$$

Our algorithm for l simply applies Theorem 3.1.1 to compute an $O(1)$-approximation to $l_{(D)}(x)$, using $\mathrm{mmc}(l_{(D)})^2 \cdot \mathrm{polylog}(n)$ bits of space. This is indeed possible because $l_{(D)}$ is clearly a symmetric norm on $\mathbb{R}^n$, and yields an $O(D)$-approximation for l, which implies D-approximation by scaling D appropriately.

It remains to bound $\mathrm{mmc}(l_{(D)})$ and show it is smaller than $\mathrm{mmc}(l)$ by factor D roughly. By Lemma 3.6.8, there is an absolute constant $\lambda > 0$ such that $\mathrm{M}_{l^{(n')}} \geq \mathrm{M}_{l^{(n)}} / (\lambda \sqrt{\log n})$ for all $n' \leq n$. Let $n^* \leq n$ be such that $\mathrm{mmc}(l) = \mathfrak{b}_{l^{(n^*)}} / \mathrm{M}_{l^{(n^*)}}$, thus $\mathrm{mmc}(l) \leq \lambda \sqrt{\log n} \, \mathrm{mc}(l^{(n)})$. Since $D \leq \mathrm{mmc}(l)$, we have

$$D \, \mathrm{M}_l \leq \lambda \sqrt{\log n} \, \mathrm{mc}(l^{(n)}) \, \mathrm{M}_l \leq \lambda \sqrt{\log n} \cdot \mathfrak{b}_l \qquad \Rightarrow \qquad \mathfrak{b}_{l_D^{(n')}} \leq \max\left(\lambda \mathfrak{b}_l / \sqrt{\log n}, \mathfrak{b}_l \right).$$

By definition of l_D, $\mathrm{M}_{l_D^{(n')}} \geq \max\left(\frac{D \, \mathrm{M}_l}{\log n}, \frac{\mathrm{M}_l}{\lambda \sqrt{\log n}} \right)$. Thus,

$$\mathrm{mmc}(l_{(D)}) \leq \frac{\lambda \log n}{D} \, \mathrm{mc}(l^{(n)}) \leq \frac{\lambda \log n}{D} \, \mathrm{mmc}(l^{(n)}).$$

We conclude that there exists a streaming algorithm computes an $O(D)$-approximation for l using $\widetilde{O}(\mathrm{mmc}(l_D)^2) = \widetilde{O}(\mathrm{mmc}(l)^2 / D^2)$ bits of space. $\square$

3.5.2 D-Approximation for General Symmetric Norms

Theorem 3.1.3 . Let l be a symmetric norm on $\mathbb{R}^n$. For every $1.1 \leq D \leq \mathrm{mmc}(l)$ there is a one-pass streaming algorithm that on input stream vector $v \in \mathbb{R}^n$ computes, with probability at least 0.99, a D-approximation to $l(v)$ and uses $(\mathrm{mmc}(l)^2/D^2) \cdot \mathrm{poly}(\log n)$ bits of space.

Proof. Let $\alpha > 1$ be a constant. Given a vector $v \in \mathbb{R}^n$ with integer coordinates, analogously to Defintion 3.3.3, define $V^\alpha = V_1^\alpha + V_2^\alpha + \ldots + V_t^\alpha$, where V_i^α is the level i vector of v with base α, and appropriate $t = O(\log n)$. For each $i \in [t]$, we define similarly $b_i^{(\alpha)}$ as the number of coordinates falling into level i. Define for each integer $1 \leq n' \leq n$, $h(\xi^{(n')}) := \min\{Dl(\xi^{(n')}), \mathfrak{b}_{l^{(n')}}\}$, and

$$h(V_i^\alpha) := h(\xi^{(b_i^{(\alpha)})}) l_2(V_i^\alpha) = \min\{Dl(V_i^\alpha), \mathfrak{b}_{l^{(b)}}\, l_2(V_i^\alpha)\}, \qquad \text{and}$$
$$h^{(\alpha)}(v) := \sum_{i \in [t]} h(V_i^\alpha).$$

We will omit the superscript α if it is clear from the context. We first claim that $h(v)$ is an $\widetilde{O}(D)$-approximation to $l(v)$. Indeed

$$l(v) \leq \alpha \sum_{i \in [t]} l(V_i) \leq \alpha \sum_{i \in [t]} \min\{Dl(V_i), \mathfrak{b}_{l^{(b_i)}}\, l_2(V_i)\} = \alpha h(v), \tag{3.5.2}$$

and by monotonicity and homogeneity of norm l

$$h(v) = \sum_{i \in [t]} h(V_i) \leq \sum_{i \in [t]} Dl(V_i) \leq Dt \max_{i \in [t]} l(V_i) \leq (\lambda D \log n) l(v),$$

where $\lambda > 0$ is a constant. Thus $h(v)$ is an $\widetilde{O}(D)$-approximation to $l(v)$.

It remains to prove that $h(v)$ can be $O(1)$-approximated using $\widetilde{O}(\mathrm{mmc}(l)^2/D^2)$ bits of space. Let $\beta = O(1/\log n)$ and

$$\beta' = O\left(\frac{D^2\beta^2}{\log^2 n\,\mathrm{mmc}(l)^2}\right).$$

Let $v \in \mathbb{R}^n$ be the streaming vector. We run algorithm `Level1` with importance parameter β', base parameter α and constant error parameter $\epsilon \in (0, 1/2)$. By Theorem 3.3.2, `Level1` is guaranteed to output a vector $\widehat{V}^{\alpha'}$ with base $\alpha' = \Theta(1)$ and with the following guarantees. Let $t' = O(\log n/\log\alpha')$, then for every $i \in [t']$, $\widehat{b}_i^{(\alpha')} \le b_i^{(\alpha')}$ and if $\widehat{V}_i^{\alpha'}$ is β'-important, then also $(1-\epsilon)b_i^{(\alpha')} \le \widehat{b}_i^{(\alpha')}$. Thus,

$$\sum_{i\in[t']} h(\widehat{V}_i^{\alpha'}) = \sum_{i\in[t']} \min\{Dl(\widehat{V}_i^{\alpha'}), \mathfrak{b}_{l(\widehat{b}_i)}\, l_2(\widehat{V}_i^{\alpha'})\} \le \sum_{i\in[t']} \min\{Dl(V_i^{\alpha'}), \mathfrak{b}_{l(b_i)}\, l_2(V_i^{\alpha'})\} = h(V^{\alpha'}).$$

We prove in Lemma 3.5.3 below that a β-contributing level of $h(v)$ (defined as $h(V_i^{\alpha'}) \ge \beta h(V^{\alpha'})$) is a β'-important level. Let $U \subset [t']$ be the set of contributing levels. Then,

$$h(\widehat{V}^{\alpha'}) \ge \sum_{i\in U} h(\widehat{V}_i^{\alpha'}) = \sum_{i\in U} \min\{Dl(\widehat{V}_i^{\alpha'}), \mathfrak{b}_{l(\widehat{b}_i)}\, l_2(\widehat{V}_i^{\alpha'})\} \ge \frac{(1-\epsilon)}{2} \sum_{i\in U} \min\{Dl(V_i^{\alpha'}), \mathfrak{b}_{l(b_i)}\, l_2(V_i^{\alpha'})\},$$

where the second inequality follows from Lemma 3.3.5 and that $\mathfrak{b}_{l_{\widehat{b}_i}} \ge \mathfrak{b}_{l_{b_i}}/2$. Indeed, let $v^* \in \mathbb{R}^{b_i}$, then we cut v^* into two pieces with roughly equal number of non-zeros $v^* = v_1^* + v_2^*$, then $l(v^*) \le l(v_1^*) + l(v_2^*) \le 2\,\mathfrak{b}_{l_{\widehat{b}_i}}$. On the other hand, $\sum_{i\notin U} h(V_i) \le t\beta h(V) \le \lambda_1 h(V)$, for some constant $\lambda_1 > 0$ that can be chosen arbitrarily small. Thus $h(\widehat{V}^{\alpha'}) \ge (1 -$

$\epsilon)(1-\lambda_1)h^{(\alpha')}(v)/2$ is a constant-factor approximation to $h(V)$. Last, by Theorem 3.3.2, `Level1` uses $\widetilde{O}(1/\beta') = \widetilde{O}(\mathrm{mmc}(l)^2/D^2)$ bits of space. □

Lemma 3.5.2. For every integers $0 < n_1 \le n_2 \le n$,

$$h(\xi^{(n_1)}) \le \frac{\lambda\sqrt{\log n}\,\mathrm{mmc}(l)}{D} h(\xi^{(n_2)}),$$

for some absoute constant $\lambda > 0$.

Proof. Since $h(\xi^{(n_1)}) = \min\left(Dl(\xi^{(n_1)}), \mathfrak{b}_{l^{(n_1)}}\right)$ and $h(\xi^{(n_2)}) = \min\left(Dl(\xi^{(n_2)}), \mathfrak{b}_{l^{(n_2)}}\right)$, then

$$\begin{aligned}
&h(\xi^{(n_1)})/h(\xi^{(n_2)}) \\
&= \max\left(\frac{\min\left(Dl(\xi^{(n_1)}), \mathfrak{b}_{l^{(n_1)}}\right)}{Dl(\xi^{(n_2)})}, \frac{\min\left(Dl(\xi^{(n_1)}), \mathfrak{b}_{l^{(n_1)}}\right)}{\mathfrak{b}_{l^{(n_2)}}}\right) \\
&\le \max\left(\lambda\sqrt{\log n}\min\left(\mathrm{mmc}(l), \frac{\mathrm{mmc}(l)}{D}\right), \min\left(\frac{\lambda D\,\mathrm{M}_{l^{(n_1)}}}{\mathfrak{b}_{l^{(n_2)}}}, \frac{\mathfrak{b}_{l^{(n_1)}}}{\mathfrak{b}_{l^{(n_2)}}}\right)\right) \\
&\le \frac{\lambda\sqrt{\log n}\,\mathrm{mmc}(l)}{D},
\end{aligned} \tag{3.5.3}$$

where the second inequality follows from Lemma 3.3.12 and Lemma 3.3.14. The last inequality uses $\mathfrak{b}_{l^{(n_1)}} \le \mathfrak{b}_{l^{(n_2)}}$ and $\lambda > 0$ is an absolute constant. □

Lemma 3.5.3. If a level i is β-contributing, i.e., $h(V_i) \ge \beta h(V)$, then

1. $b_i \ge \frac{\lambda D^2\beta^2}{\log^2 n\,\mathrm{mmc}(l)^2}\sum_{j>i} b_j$;
2. $b_i\alpha^{2i} \ge \frac{\lambda D^2\beta^2}{\log^2 n\,\mathrm{mmc}(l)^2}\sum_{j\le i} b_j\alpha^{2j}$,

for some constant $\lambda > 0$.

Proof. The proof is similar to that of Lemma 3.3.8 and that of Lemma 3.3.15. Since level i is β-contributing, we have

$$h(V_i) \geq \beta \sum_{j \in [t]} h(V_j).$$

Let $j^* = \text{argmax}_{j>i} b_j$. We can assume $b_i \leq b_{j^*}$ since otherwise $b_i \geq \sum_{j>i} b_i/t$. Thus, by Lemma 3.5.2

$$h(V_i) \geq \beta h(V_{j^*}) \Rightarrow \sqrt{b_i} \geq \frac{D\beta}{\sqrt{\lambda' \log n}\,\text{mmc}(l)} \sqrt{b_{j^*}} \Rightarrow b_i \geq \frac{D^2\beta^2}{\lambda' t \log n\,\text{mmc}(l)^2} \sum_{j>i} b_j.$$

where $\lambda' > 0$ is an absolute constant.

For the second inequality, let $j' := \text{argmax}_{j \leq i} \sqrt{b_j}\alpha^j$. We proceed by separating into two cases. First, if $b_i \geq b_{j'}$ then the lemma follows easily by

$$b_i \alpha^{2i} \geq b_{j'} \alpha^{2j'} \geq \frac{\sum_{j \leq i} b_j \alpha^{2j}}{t}.$$

The second case is when $b_i < b_{j'}$,

$$\alpha^i \sqrt{b_i} h(\xi^{(i)}) = h(V_i) \geq \beta h(V) \geq \beta h(V_{j'}) = \beta \alpha^{j'} \sqrt{b_{j'}} h(\xi^{(j')}).$$

By Lemma 3.5.2, we get

$$\alpha^i \sqrt{b_i} \geq \beta \alpha^{j'} \sqrt{b_{j'}} \frac{h(\xi^{(j')})}{h(\xi^i)} \geq \frac{D\beta \sqrt{b_{j'}} \alpha^{j'}}{\sqrt{\lambda'' \log n}\,\text{mmc}(l)},$$

where $\lambda'' > 0$ is an absolute constant. Squaring the above and observing that

$$b_i \alpha^{2i} \geq \frac{D^2 \beta^2}{\lambda'' t \log n \, \mathrm{mmc}(l)^2} \sum_{j \leq i} b_j \alpha^{2j},$$

the proof is complete. □

3.6 Applications & Examples

3.6.1 The Top-k Norm $\Phi_{(k)}$

The *top-k* norm on $\mathbb{R}^n$ is simply the sum of the k largest coordinates in absolute value, formally, $\Phi_{(k)}(x) := \sum_{i=1}^{k} |x|_{[i]}$, where $|x|_{[1]} \geq \ldots \geq |x|_{[n]}$ are the coordinates ordered by non-increasing absolute value. It is known (see e.g. [81, Exer. IV.1.18]) that the dual norm of $\Phi_{(k)}$ is $\Phi'_{(k)}(x) := \max\{l_\infty(x), l_1(x)/k\}$. We can understand the streaming space complexity of such a norm l by comparing the maximum and the median of such a norm over S^{n-1}, which is an easy calculation, and then applying Theorems 3.1.1 and 3.1.2.

Theorem 3.6.1. There are absolute constants $\lambda_1, \lambda_2 > 0$ such that for all $k = 1, \ldots, n$,

$$\lambda_1 \sqrt{\frac{n}{k \log n}} \leq \mathrm{mmc}(\Phi_{(k)}) \leq \lambda_2 \sqrt{\frac{n}{k}}, \qquad \text{and } \lambda_1 \sqrt{\frac{k}{\log(k)+1}} \leq \mathrm{mmc}(\Phi'_{(k)}) \leq \lambda_2 \sqrt{k}.$$

The above inequalities are existentially tight, by considering the cases $k = 1$ and $k = n$. To prove this theorem, we will need the next two lemmas. They both assume $1 \leq k \leq n$.

Lemma 3.6.2. For all $x \in \mathbb{R}^n$, $\sqrt{\frac{k}{n}}\, l_2(x) \leq \Phi_{(k)}(x) \leq \sqrt{k}\, l_2(x)$.

We remark that the second inequality above is tight for $x = \xi^{(k)}$.

Proof. Fix $x \in \mathbb{R}^n$. We use Cauchy-Schwarz

$$\sum_{i=1}^{k} |x|_{[i]} \le \sqrt{k} \left(\sum_{i=1}^{k} |x|_{[i]}^2 \right)^{1/2} \le \sqrt{k} \left(\sum_{i=1}^{n} x_{[i]}^2 \right)^{1/2}.$$

For the second inequality, we use monotonicity of l_p-norms

$$\sum_{i=1}^{k} |x|_{[i]} \ge \left(\sum_{i=1}^{k} |x|_{[i]}^2 \right)^{1/2} \ge \left(\frac{k}{n} \sum_{i=1}^{n} |x|_{[i]}^2 \right)^{1/2}. \qquad \square$$

Lemma 3.6.3. $\frac{\lambda_1 k}{\sqrt{n}} \le \mathrm{M}_{\Phi_{(k)}} \le \frac{\lambda_2 k \sqrt{\log n}}{\sqrt{n}}$ for some absolute constants $\lambda_1, \lambda_2 > 0$.

Proof. For the first inequality, $\Phi_{(k)}(x) \ge \frac{k}{n} \sum_{i=1}^{n} |x|_{[i]} = \frac{k}{n}\, l_1(x)$. Therefore, $\mathrm{M}_{\Phi_{(k)}} \ge \frac{k}{n} \mathrm{M}_{l_1} \ge \lambda_1 k/\sqrt{n}$ for some absolute constant $\lambda_1 > 0$. For the second inequality, $\Phi_{(k)}(x) \le k l_\infty(x)$, and thus $\mathrm{M}_{\Phi_{(k)}} \le k \mathrm{M}_{l_\infty} \le \frac{\lambda_2 k \sqrt{\log n}}{\sqrt{n}}$ for some absolute constant $\lambda_2 > 0$. $\square$

Proof of Theorem 3.6.1. To bound $\mathrm{mmc}(\Phi_{(k)})$, consider first $n' \ge k$, then by a direct calculation,

$$\frac{\sqrt{k}}{(\lambda_2 k \sqrt{\log n'}/\sqrt{n'})} \le \frac{\mathfrak{b}_{\Phi_{(k)}^{(n')}}}{\mathrm{M}_{\Phi_{(k)}^{(n')}}} \le \frac{\sqrt{k}}{(\lambda_1 k/\sqrt{n'})}.$$

For $n' \le k$, we have $\Phi_{(k)}(x) = l_1(x)$ for all $x \in \mathbb{R}^{n'}$, and we know that $\mathrm{mmc}(l_1)$ is a constant. The first part of the theorem follows.

To bound $\mathrm{mmc}(\Phi'_{(k)})$, consider first the case $n' \ge k$. For all $x \in \mathbb{R}^{n'}$ we have $\Phi'_{(k)}(x) \ge$

$l_1(x)/k$, thus $\mathrm{M}_{\Phi'^{(n')}_{(k)}} = \Omega(\sqrt{n'}/k)$. In addition, $\mathfrak{b}_{\Phi'^{(n')}_{(k)}} \leq \max\{1, \sqrt{n'}/k\}$, and thus

$$\mathfrak{b}_{\Phi'^{(n')}_{(k)}} / \mathrm{M}_{\Phi'^{(n')}_{(k)}} \leq \sqrt{k}.$$

Consider now the case $n' \leq k$. For all $x \in \mathbb{R}^{n'}$, we have $\Phi'_{(k)}(x) = l_\infty(x)$, and thus, $\mathfrak{b}_{\Phi'^{(n')}_{(k)}} / \mathrm{M}_{\Phi'^{(n')}_{(k)}} = \Theta(\sqrt{n'/\log n'})$. We conclude that $\mathrm{mmc}(\Phi'_{(k)}) = \Omega(\sqrt{k/(\log k + 1)})$. □

3.6.2 Q-Norms and Q'-Norms

A norm $l : \mathbb{R}^n \to \mathbb{R}$ is called a *Q-norm* if there exists a symmetric norm $\Phi : \mathbb{R}^n \to \mathbb{R}$ such that

$$\forall x \in \mathbb{R}^n, \qquad l(x) = \Phi(x^2)^{1/2},$$

where $x^p = (x_1^p, x_2^p, \ldots, x_n^p)$ denotes coordinate-wise p-th power. A norm $l' : \mathbb{R}^n \to \mathbb{R}$ is called a *Q'-norm* if its dual norm, which is given by $l(x) = \sup\{\frac{\langle x,y\rangle}{l'(y)} : y \neq 0\}$, is a Q-norm.

We can show that every Q'-norm can be approximated using polylogarithmic space, by bounding $\mathfrak{b}_{l'} / \mathrm{M}_{l'}$ and then applying Theorem 3.1.1, as follows.

Theorem 3.6.4. For every Q'-norm $l' : \mathbb{R}^n \to \mathbb{R}$, $\mathrm{mmc}(l') = O(\log n)$.

Corollary 3.6.5 (Streaming Complexity of Q'-Norms)**.** Every Q'-norm $l' : \mathbb{R}^n \to \mathbb{R}$ can be $(1+\epsilon)$-approximated by a one-pass streaming algorithm that uses $\mathrm{poly}(\log(n)/\epsilon)$ space.

The proof of Theorem 3.6.4 will follow by establishing the four lemmas below. It builds on the machinery developed in Section 3.3 to compare the median of l to $l(\xi^{(n')})$, where $\xi^{(n')}$ is the l_2-normalized all-ones vector of dimension n'.

Lemma 3.6.6. Let $l : \mathbb{R}^n \to \mathbb{R}$ be a Q-norm, and let $0 < n' \leq n$. Then

$$l(\xi^{(n')}) \geq l(\xi^{(n)})/2.$$

Proof. Write $n = qn' + r$, where $r < n'$ is the remainder. Then

$$\xi^{(n)} = \left(\sqrt{\tfrac{n'}{n}} (\underbrace{\xi^{(n')}, \ldots, \xi^{(n')}}_{q \text{ times}}), \sqrt{\tfrac{r}{n}} \xi^{(r)} \right).$$

By monotonicity of symmetric norms and the triangle inequality,

$$l(\xi^{(n)}) \leq l\left(\sqrt{\tfrac{n'}{n}} (\underbrace{\xi^{(n')}, \ldots, \xi^{(n')}}_{q \text{ times}}) \right) + l\left(\sqrt{\tfrac{r}{n}} \xi^{(r)} \right) \leq 2l\left(\sqrt{\tfrac{n'}{n}} (\underbrace{\xi^{(n')}, \ldots, \xi^{(n')}}_{q \text{ times}}) \right).$$

We can write $l(x) = \Phi(x^2)^{1/2}$ for some symmetric norm Φ. Thus, by the triangle inequality,

$$\begin{aligned} l\left(\sqrt{\tfrac{n'}{n}} (\underbrace{\xi^{(n')}, \ldots, \xi^{(n')}}_{q \text{ times}}) \right) &= \sqrt{\tfrac{n'}{n}}\ \Phi\left((\xi^{(n')})^2, \ldots, (\xi^{(n')})^2 \right)^{1/2} \\ &\leq \sqrt{\tfrac{n'}{n}} \left(q\ \Phi((\xi^{(n')})^2) \right)^{1/2} \leq l(\xi^{(n')}). \end{aligned}$$

□

Lemma 3.6.7. Let $l : \mathbb{R}^n \to \mathbb{R}$ be a Q-norm, then for all $x \in \mathbb{R}^n,\ l(x) \leq l_2(x)$.

Proof. Let $l(x) = \Phi(x^2)^{1/2}$ for some symmetric norm Φ. By Lemma 3.2.2, $\Phi(x) \leq l_1(x)$ and therefore $l(x) = (\Phi(x^2))^{1/2} \leq l_1(x^2)^{1/2} = l_2(x)$. □

The next lemma can be viewed as a complement of Lemma 3.3.14 (monotonicity of the

median) for the special case of Q-norms.

Lemma 3.6.8. Let $l : \mathbb{R}^n \to \mathbb{R}$ be a Q-norm, and let $0 < n' \le n$ be an integer. Then

$$\mathrm{M}_{l^{(n)}} \le \lambda \sqrt{\log n}\, \mathrm{M}_{l^{(n')}}$$

for some absolute constant $\lambda > 0$.

Proof. By Lemmas 3.3.12 and 3.6.6, we can find absolute constant $\lambda_1, \lambda_2 > 0$ such that

$$\lambda_1 \mathrm{M}_{l^{(n)}} / \sqrt{\log n} \le l(\xi^{(n)}) \le 2l(\xi^{(n')}) \le 2\lambda_2 \mathrm{M}_{l^{(n')}}. \qquad \square$$

Now we show that a Q-norm achieves roughly the minimum at $\xi^{(n)}$.

Lemma 3.6.9 (Flat Minimum)**.** Let $l : \mathbb{R}^n \to \mathbb{R}$ be a Q-norm. Then

$$\forall x \in S^{n-1}, \qquad l(\xi^n) \le 6\sqrt{\log n}\; l(x).$$

Proof. Set $\alpha := 1/2$ and fix a vector $x \in S^{n-1}$. We permute its coordinates and write $|x| = (V_1; V_2; \ldots; V_t; V')$, where $V_i = \{|x_j| : \alpha^j < |x_j| \le \alpha^{j-1}\}$ for $i = 1, \ldots, t = \log n$, and $V' = \{|x_j| : |x_j| \le 1/n\}$. Let $b_i = |V_i|$. Since $l_2(x) = 1$,

$$1 = l_2(x)^2 \le \sum_{i=1}^{t} b_i \alpha^{2(i-1)} + 1/n.$$

Thus, there exists $i \le t$ for which $|V_i|\alpha^{2(i-1)} \ge \frac{1}{2t}$, and together with Lemma 3.6.6,

$$l(x) \ge l(V_i) \ge \sqrt{b_i}\alpha^i \; l(\xi^{(b_i)}) \ge \sqrt{\tfrac{\alpha^2}{2t}} \; l(\xi^{(b_i)}) \ge \sqrt{\tfrac{1}{8t}} \; l(\xi^{(n)})/2. \qquad \square$$

Proof of Theorem 3.6.4. Let l be the Q-norm which is dual to l'. By Lemma 3.6.9, $\forall x \in \mathbb{R}^n$, $l_2(x) \le 6\sqrt{\log n}/l(\xi^{(n)}) \cdot l(x)$, which implies, using Fact 3.2.3, that $\mathfrak{b}_{l'} \le 6\sqrt{\log n}/l(\xi^{(n)})$. By Fact 3.2.4 and Lemma 3.3.12, we know that $1/\operatorname{M}_{l'} \le \operatorname{M}_l \le l(\xi^{(n)})\sqrt{\log n}/\lambda_1$. The theorem follows by putting the two bounds together. $\square$

3.7 The Level Algorithm

In this section we prove Theorem 3.3.2 by presenting the level algorithm and analyzing its performance. The algorithm follows the ideas used by Indyk and Woodruff.[15] for approximating frequency moments. As their paper focuses on the specific problem of l_p-norms, its analysis is more specialized, with parameters chosen based on properties of l_p-norms, and it is not immediate to see how to generalize/modify it to approximate all symmetric norms. For an easier statement of our upper bound, and also for completeness, we modify their algorithm to output level vectors (instead of the value of l_p norm) and repeat the analysis accordingly.

To present the high level idea, we first present a two-pass algorithm, and then modify it to be a one-pass algorithm. The two-pass algorithm is shown in Algorithm 5. Note that we assume full randomness, i.e., the algorithm has unlimited access to random bits. We can

reduce the number of bits needed to $O(\log n)$ by following Indyk's method,[16] which uses Nisan's pseudorandom generator,[99] and this impacts the space complexity by an $O(\log n)$ factor.

3.7.1 Two-Pass Algorithm

The purpose of this section is to prove the following theorem.

Theorem 3.7.1. There is a streaming two pass algorithm `TwoPassLevelCounts`, that given input stream $\mathcal{S}$ with frequency vector v, level base $\alpha > 1$, importance $\beta > 0$, precision $\epsilon > 0$ and error probability δ, output a list $(\widehat{b}_1, \widehat{b}_2, \ldots, \widehat{b}_t)$, where $t = \log n / \log \alpha$, such that,

- for all $i \in [t]$, $\widehat{b}_i \leq b_i$;
- if i is a β-important level, then $\widehat{b}_i \geq (1-\epsilon) b_i$,

with probability at least $1 - \delta$, using space $O\left(\frac{\log^6 n}{\beta \epsilon^4} \frac{\log^2(1/\delta)}{\log \alpha} \log \frac{n^2}{\delta}\right)$.

The high level idea of the two pass algorithm is as follows. For each $\phi = 0, 1, \ldots, O(\log n)$, we select a random subset of $[n]$ where each item is included independently with probability $2^{-\phi}$. This gives us $O(\log n)$ substreams, where each is defined by restricting the original stream to only include updates from one of the random subsets. What we will prove is that if level i is important, then a random sample of the items in level i will appear as heavy hitters in one of the substreams. Hence, we can find them with a CountSketch.[67] The entire sketch, that is the subsampling combined with CountSketch, is presented in Algorithm 4, SampleLevel. Finding the largest ϕ such that some item from level i appears in the substream gives us the estimate 2^{ϕ} for the size of that level, but this estimate will not

be accurate enough for our purposes.

To get an accurate estimate of the level sizes we repeat the above procedure to identify heavy hitters $R = O(\epsilon^{-2}\log^2 n)$ times independently in parallel. Next, on the second pass, we determine the exact frequency of each of the items identified during the first pass and thus correctly identify its level. The final estimate for the size of level i is derived by considering the largest ϕ such that at least a $\Omega(1/\log n)$ fraction among the repetitions with sampling probability $2^{-\phi}$ contained an item in level i. The entire level vector approximation procedure is Algorithm 5.

Algorithm 4 $\texttt{SampleLevel}(\mathcal{S}, n, \beta, \epsilon, \delta, \Phi, R)$, sketch of the frequency vector by subsampling and finding β heavy hitters

Input: stream $\mathcal{S}$, $\beta > 0$, $\epsilon > 0$, $\delta > 0$, $\Phi > 0$, R.
Output: Collection of maps $\{D^r_\phi \mid \phi \in [0,\Phi], r \in [0,R]\}$.
For each $r \in [0,R]$ and $\phi \in [0,\Phi]$ generate a substream $\mathcal{S}^r_\phi$ by sampling each $i \in [n]$ with probability $p_\phi = 2^{-\phi}$, independently, and including all updates to i
Let $D^r_\phi = \texttt{CountSketch}(\mathcal{S}^r_\phi, \epsilon, \beta, \delta)$.
Return $\{D^r_\phi \mid \phi \in [0,\Phi], r \in [0,R]\}$.

Algorithm 5 $\texttt{TwoPassLevelCounts}(\mathcal{S}, n, \alpha, \beta, \epsilon, \delta)$, a two-pass algorithm for approximating the level vector

Input: stream $\mathcal{S}$, $\alpha > 1$, $\beta > 0$, $\epsilon > 0$ and $\delta > 0$
Output: $(\widehat{b}_0, \widehat{b}_1, \widehat{b}_2, \widehat{b}_3 \ldots \widehat{b}_t)$
Initialization: Let $\Phi \leftarrow \log n$, $R \leftarrow \Theta\left(\frac{\log(1/\delta)\log^2 n}{\epsilon^2}\right)$, $\epsilon' \leftarrow \Theta(\epsilon)$
First Pass:
$\widetilde{\mathcal{T}} \leftarrow \texttt{SampleLevel}(\mathcal{S}, n, O\left(\frac{\beta}{t\log(1/\delta)}\right), \epsilon', \delta/n, \Phi, R)$
Second Pass:
$\mathcal{T} \leftarrow$ the exact frequencies of all maps in $\widetilde{\mathcal{T}}$
Estimation Stage:
For each $\phi \in [\Phi]$ and each $i \in [t]$, let $A_{\phi,i} \leftarrow |\{r \mid \exists i \in D^r_\phi, \alpha^{i-1} < |D^r_\phi[i]| \le \alpha^i\}|$
For each $i \in [t]$, set $q_i \leftarrow \max_{\phi\in[0,\Phi]}\{\phi \mid A_{\phi,i} \ge \frac{R\log\frac{1}{\delta}}{100\log n}\}$
If q_i does not exist then $\widehat{\eta}_i \leftarrow 0$, else $\widehat{\eta}_i \leftarrow A_{q_i,i}/(R(1+\epsilon'))$
If $\widehat{\eta}_i = 0$ then $\widehat{b}_i \leftarrow 0$, else $\widehat{b}_i \leftarrow \frac{\log(1-\widehat{\eta}_i)}{\log(1-2^{-q_i})}$

Heavy hitter algorithm: For convenience, we define a *map* data structure, D, as a set of pairs from $[n] \times \mathbb{Z}$ with the property that for each $i \in [n]$ there is at most one pair $(i, \cdot)$ in D. For any $i \in [n]$, we say $i \in D$ if there is a pair (i, z) in D and when $i \in D$ we denote $D[i] = z$ as the value paired with i.

Define $F_2^{\geq k}(\mathcal{S}) := \sum_{i=k+1}^{n} |v_{[i]}|^2$, where $|v_{[0]}| \geq |v_{[1]}| \ldots \geq |v_{[n]}|$ are the coordinates of the frequency vector in decreasing order with ties broken arbitrarily.

Definition 3.7.2. We call a map D a (β, ϵ)-cover of the steam $\mathcal{S}$, if

- if for some $i \in [n]$ such that $|v_i|^2 \geq \beta F_2^{\geq \lfloor 1/\beta \rfloor}(\mathcal{S})$ then $i \in D$;
- for every $j \in D$, $|v_j| \leq D[j] \leq (1+\epsilon)|v_j|$;

We omit $\mathcal{S}$ if it is clear from context.

The purpose of Algorithm 4 is to find a (β, ϵ)-cover for each of $O(\log n)$ substreams. We call ϵ the *precision parameter*, β the *heaviness parameter*, and δ the *error rate*[2].

Theorem 3.7.3 ([67])**.** There is a one pass streaming algorithm `CountSketch`$(\mathcal{S}, \epsilon, \beta, \delta)$ that, for any input stream $\mathcal{S}$ of universe $[n]$, with frequency vector $v = (v_1, v_2, \ldots, v_n)$, outputs a map D such that, with probability at least $(1 - \delta)$,

- D is a (β, ϵ)-cover of $\mathcal{S}$ and
- $|D| \leq 2/\beta$.

CountSketch uses $O(\frac{1}{\beta\epsilon^2} \log \frac{n}{\delta} \log n)$ bits of space.

Consider the algorithm `SampleLevel`, we define $\mathcal{E}_\phi^r$ as the event that the (ϕ, r) instance

[2]We asume $\delta < \epsilon$ for the following analysis.

of `CountSketch` outputs a (β, ϵ)-covers for the substream $\mathcal{S}^r_\phi$. By Theorem 4.3.5, we have $\Pr[\mathcal{E}^r_\phi] \geq 1 - \frac{\delta}{n}$. Note that $\mathcal{E}^r_\phi$ is independent of $\mathcal{S}^r_\phi$. To simplify analysis, we will assume that the output of each `CountSketch` is correct. Our choice of the error rate δ will be such that this happens with probability near 1. For the following analysis, we consider Algorithm `TwoPassLevelCounts` and assume $\epsilon = 1/\text{polylog}n, \delta = 1/\text{poly}(n), \beta = 1/\text{polylog}n$.

Detectability of Levels:

Definition 3.7.4. A level i of stream $\mathcal{S}$ is β-*detectable*, if

$$\alpha^{2i} \geq \beta F_2^{\geq \lfloor 1/\beta \rfloor}(\mathcal{S}).$$

We omit $\mathcal{S}$ if it is clear from the context.

If a level i is β-detectable then a `CountSketch`, with heaviness β, will include B_i in its output. The following lemma is about the detectability of an important level.

Lemma 3.7.5. Suppose substream $\mathcal{S}'$ is obtained by subsampling the original stream with probability p. Assume $\alpha \leq 2$. If i is a β-important level, then for any $\lambda > t$, with probability at least $1 - t\exp\left[-\frac{\lambda p b_i}{t\beta}\right]$, level i is $\frac{\beta}{\lambda p b_i}$-detectable. In particular, if $pb_i = O(1)$, then with probability at least $1 - t\exp\left[-\Omega\left(\frac{\lambda}{t\beta}\right)\right]$, level i is $\frac{\beta}{\lambda}$-detectable.

Proof. Let V' be the new frequency vector the substream. Let $(b'_0, b'_1, \ldots)$ be the new level sizes. Thus, for all $j \in [t]$,

$$E(b'_j) = pb_j. \tag{3.7.1}$$

By the definition of important level

$$E[b'_i] \geq \beta E[\sum_{j>i} b'_j]$$

and

$$E[b'_i \alpha^{2i}] \geq \beta E[\sum_{j \leq i} b'_j \alpha^{2j}].$$

To have level i detectable, it has to be among the top $\lambda p b_i/\beta$ elements, thus $\sum_{j>i} b'_j \leq \lambda p b_i/\beta$. The probability this does not happen is, by Chernoff's bound,

$$\Pr[\sum_{j>i} b_j > \frac{\lambda p b_i}{\beta}] \leq \exp[-\Omega(\frac{\lambda p b_i}{\beta})].$$

On the otherhand, for level i to be detectable $\alpha^{2i} \geq \frac{\beta}{\lambda p b_i} \sum_{j \leq i} b'_j \alpha^{2j}$. Thus, the complement occurs with probability,

$$\begin{aligned}\Pr\left[\frac{\beta}{\lambda p b_i}\sum_{j\leq i} b'_j\alpha^{2j} > \alpha^{2i}\right] &\leq \Pr\left[\exists j, \quad b'_j\alpha^{2j} \geq \frac{\lambda p b_i \alpha^{2i}}{t\beta}\right] \\ &\leq \sum_j \Pr[b'_j\alpha^{2j} \geq \frac{\lambda p b_i \alpha^{2i}}{t\beta}].\end{aligned}$$

By Chernoff's bound and because $E(b'_j\alpha^{2j}) \leq p b_i \alpha^{2i}$,

$$\Pr\left[\frac{\beta}{\lambda p b_i}\sum_{j\leq i} b'_j\alpha^{2j} > \alpha^{2i}\right] \leq t\exp\left[-\Omega\left(\frac{\lambda p b_i \alpha^{2i}}{\alpha^{2j} t\beta}\right)\right] \leq t\exp\left[-\Omega\left(\frac{\lambda p b_i}{t\beta}\right)\right]. \qquad \square$$

Subsample the important levels: Let us define $\eta_{i,\phi} := 1 - (1 - p_\phi)^{b_i}$, which is the probability that at least one element from B_i is sampled when the sampling probability is $p_\phi = 2^{-\phi}$. Set $\lambda = \Theta(t \log \frac{1}{\delta})$. Let $\eta'_{i,\phi}$ be the probability that an element from B_i is contained in D^1_ϕ, which is the same as the probability that it is contained in D^r_ϕ for any other r.

Lemma 3.7.6. For any level i we have $\eta'_{i,\phi} < \eta_{i,\phi}$. Also, assume $\delta < \epsilon$, if level i is a β-important level and $p_\phi b_i = O(1)$, then $\eta'_{i,\phi} \geq (1 - \Theta(\epsilon))\eta_{i,\phi}$.

Proof. For an element to be in D^r_ϕ it must first be sampled, thus $\eta'_{i,\phi} \leq \eta_{i,\phi}$. On the other hand, with probability at least $1 - t \exp\left[-\Omega(\frac{\lambda p b_i}{t\beta})\right] = 1 - O(\delta)$, level i is $\frac{\beta}{\lambda p b_i}$-detectable. Note that $\frac{\beta}{\lambda p b_i} = \Theta(\frac{\beta}{t \log(1/\delta)})$. Thus with probability at least $\eta_{i,\phi}(1-\Theta(\delta)) \geq \eta_{i,\phi}(1-O(\epsilon)))$ an element from B_i is sampled and the element is reported by `CountSketch`. □

We now show that $\widehat{\eta}_i$, at Line 11 of Algorithm 5, is a good estimator for η'_{i,q_i}.

Lemma 3.7.7. At Line 11, with probability at least $1 - \delta^{\Omega(\log n)}$, for all $i \in [t]$, if $\widehat{\eta}_i \neq 0$ then $(1 - O(\epsilon))\eta'_{i,q_i} \leq \widehat{\eta}_i \leq \eta'_{i,q_i}$. The probability is taken over the probability space of all the random bits chosen for the algorithm.

Proof. If $\widehat{\eta}_i \neq 0$, then $A_{q_i,i} \geq \gamma = \frac{R \log \frac{1}{\delta}}{100 \log n}$. For a particular i, since the sampling process is independent for each $r \in [R]$, the claim is that $E(A_{q_i,i}) = R\eta_{i,q_i} = \Omega(\gamma)$. This holds because otherwise, by a Chernoff bound, $\Pr[A_{q_i,i} \geq \gamma] = o(\delta^{\Omega(\log n)})$. Therefore by a Chernoff bound,

$$\Pr[|A_{q_i,i} - R\eta'_{i,q_i}| \geq \epsilon R \eta_{i,q_i}] \leq \exp(-\Omega(\epsilon^2 \gamma)) = \delta^{\Omega(\log(n))}.$$

Since $t = \text{polylog}(n)$, by union bound, $(1-\epsilon')\eta_{i,q_i} \le A_{q_i,i}/R_i \le (1+\epsilon')\eta_{i,q_i}$ happens for all $i \in [t]$ with probability at least $1 - \delta^{\Omega(\log(n))}$. Thus the lemma is proved. □

Lemma 3.7.8. At Line 11, if level i is important, then with probability at least $1-\delta^{\Omega(\log n)}$, the maximizer q_i is well defined. The probability is taken over the probability space of all the random bits chosen for the algorithm.

Proof. By Lemma 3.7.6 we know when $p_i b_i = O(1)$ level i is at least $\Omega(\frac{\beta}{t\log(1/\delta)})$-detectable. On the other hand, consider the case when $p_i = 2^{-\phi_0} = \frac{1}{b_i}$, we have $\eta_{i,\phi_0} = 1-(1-p_i)^{b_i} \ge 1/e$. Thus $E(A_{\phi_0,i}) \ge R/e$, thus by Chernoff's bound,

$$\Pr[A_{\phi_0,i} \le \frac{R\log 1/\delta}{100\log n}] \le \exp\left[-\Omega\left(\frac{R\log 1/\delta}{\log n}\right)\right] \le \delta^{\Omega(\log n)}.$$

Thus, there exists a $q_i \ge \phi_0$ with probability at least $1-\delta^{\Omega(\log n)}$. Since there are at most $t = \text{polylog}(n)$ important levels, with probability at least $1-\delta^{\Omega(\log n)}$, the corresponding value of q_i is well defined for all important levels. □

From probability estimation to size of the level: Now we show that the conversion from $\widehat{\eta}_i$ to $\widehat{b}_i$ gives a good approximation to b_i.

Lemma 3.7.9. At line 12 of Algorithm 5. Suppose $q_i \ge 1$, $\epsilon \le 1/2$, and n is sufficiently large. If $\widehat{\eta}_i \le \eta_{i,q_i}$ then $\widehat{b}_i \le b_i$. If $\widehat{\eta}_i \ge (1-\epsilon)\eta_{i,q_i}$ then $\widehat{b}_i \ge (1-O(\epsilon))b_i$.

Proof. Note that

$$b_i = \frac{\log(1-\eta_{i,q_i})}{\log(1-2^{-q_i})},$$

is a increasing function of η_{i,q_i}. Thus if $\widehat{\eta}_i \leq \eta_{i,q_i}$, then $\widehat{b}_i \leq b_i$. On the other hand, if $\widehat{\eta}_i \geq (1 - O(\epsilon))\eta_{i,q_i}$ we have

$$\widehat{b}_i \geq b_i + \frac{\epsilon' \eta_{i,q_i}}{(1 - \eta_{i,q_i}) \log(1 - 2^{-q_i})} \geq b_i - O(\epsilon) b_i. \qquad \square$$

Full Proof:

Proof of Theorem 3.7.1. Define the event $\mathcal{E}_2$ that for all important levels, q_i is well defined. Define event $\mathcal{E}_3$ that for all $i \in [t]$, if $\widehat{\eta}_i > 0$ then $(1 - O(\epsilon))\eta'_{q_i,i} \leq \widehat{\eta}_i \leq \eta'_{q_i,i}$. By Lemma 3.7.7 and 3.7.8, we have that $\Pr[\mathcal{E}_2 \cap \mathcal{E}_3] \geq 1 - \delta^{O(\log n)}$. When the output of every CountSketch is correct, it follows from lemmas 3.7.6, 3.7.7, and 3.7.9 the algorithm outputs an approximation to the level vector meeting the two criteria.

`CountSketch` is used a total of ΦR times, each with error probability at most δ/n. By a union bound, the failure probability is at most $(\text{polylog}n)\delta/n = o(\delta)$. Therefore, the total failure probability of the algorithm is at most $1 - o(\delta)$.

The last step of the proof is to bound the space used by the algorithm. For each instance of the `CountSketch`, the space used is $O(\frac{1}{\epsilon^2} \frac{t \log(1/\delta)}{\beta} \log \frac{n^2}{\delta} \log n)$. There are ΦR instances. Finally, we can reduce the number bits need to $O(\log n)$ by using Nisan's pseudorandom generator[99] with Indky's method,[16] which impacts the storage by a $O(\log n)$ factor. To store the level vector, it requires t counters. Therefore, the total space used is,

$$O\left(\frac{\log^6 n}{\beta \epsilon^4} \frac{\log^2(1/\delta)}{\log \alpha} \log \frac{n^2}{\delta} + \frac{\log^2 n}{\log \alpha}\right) \tag{3.7.2}$$

bits of memory. □

3.7.2 One-Pass Algorithm

In this section, following Indyk-Woodruff 's[15] approach, we show that we can convert the two-pass algorithm to a one pass algorithm by randomizing the boundary. The randomizing scheme works by changing $\alpha = 1+\gamma$ to $\alpha' = 1+x\gamma$, where x is chosen uniformly at random from $[1/2, 1]$. Note that in,[15] they randomize the boundary by changing the level boundaries to $x\alpha^i$, we will verify that their proof still works if choose our boundary randomization scheme. Here we assume that the algorithm works for real numbers. The full approach in[15] proves that the real number can be represented by first few bits of the real number x and yeilds no precision loss.

We refer our algorithm as `Level1`. The idea is to remove the second pass in Algorithm `TwoPass- LevelCounts`, and just use the approximated values $D[j]$s. Note that in `TwoPassLevelCounts`, we require the second pass to measure the exact frequencies, so that we can accurately decide which level the sampled frequency belongs to. The task of deciding levels is to test whether $|v_j| \geq \alpha^i$. However, for each returned map D, since $|v_j| \leq D[j] \leq (1+\epsilon)|v_j|$, if the boundary $\alpha^i \in [|v_j|, (1+\epsilon)|v_j|]$, then $D[j] \geq \alpha^i$ may not imply $|v_j| \geq \alpha^i$.

Following,[15] after the running of `SampleLevel`, for a returned map D, for each $j \in D$, we claim level $w = \lceil \log D[j] / \log \alpha' \rceil$ is detected. Thus whether a level w is detected, is determined by the *first* $j \in [m]$ that $D[j]$ falls in interval $(\alpha'^{w-1}, \alpha'^{w}]$. Denote $v = |v_j|$ as

the actual frequency. We test whether the event $\mathcal{E}_w : \alpha'^{w-1} \geq D[j]/(1+\epsilon)$ happens. If true, we discard the returned map D. If this is not the case, we know that $|v| \in (\alpha'^{w-1}, \alpha'^{w}]$, and therefore w is the correct classification of $|v|$.

First note that if $\alpha < 2$, then the case $w = 1$ is always a good case since $D[j] \in (1, \alpha'] \Rightarrow v \in (1/(1+\epsilon), \alpha']$, which is only possible when $|v| = 1$. Then we can combine the case $w = 1$ and $w = 0$. When $w > 1$, assuming the precision parameter of `CountSketch` is ϵ', we bound the probability of the bad cases, which imply,

$$
\begin{aligned}
&(1-\epsilon')|v| \leq (1 + x\gamma)^{w-1} \leq (1+\epsilon')|v| \\
&\quad \Rightarrow \log(1-\epsilon') + \log|v| \leq (w-1)\log(1+x\gamma) \leq \log|v| + \log(1+\epsilon') \\
&\quad \Rightarrow \frac{\log|v|}{\gamma} - \frac{\epsilon'}{\gamma} \leq (w-1)x \leq \frac{\log|v|}{\gamma} + \frac{\epsilon'}{\gamma} \\
&\quad \Rightarrow \frac{\log|v|}{(w-1)\gamma} - \frac{\epsilon'}{(w-1)\gamma} \leq x \leq \frac{\log|v|}{(w-1)\gamma} + \frac{\epsilon'}{(w-1)\gamma}
\end{aligned} \tag{3.7.3}
$$

Since $w > 1$. Thus $\Pr[\mathcal{E}_w] \leq \frac{4\epsilon'}{(w-1)\gamma} \leq \frac{4\epsilon'}{\gamma}$. We can choose the precision parameter as, $\epsilon' = \frac{\gamma^2}{8\log n}$. And there are at most $t' = \log n / \log \alpha' = O(\log n/\gamma)$ detected levels. $\mathcal{E}_w$ happens for any of the detected levels $w \in [t']$ with probability at most,

$$
\frac{\log n}{\log \alpha'} \frac{\epsilon'}{\gamma} \leq \frac{\epsilon' \log n}{\gamma^2} = \frac{1}{2}.
$$

Thus, by parallel repeating the `SampleLevel` algorithm $\log \frac{n^2}{\delta}$ times, with probability at list $1 - O(\delta/n^2)$, there is a good map D in every $\log \frac{n^2}{\delta}$ many returned maps. Thus, with probability at least $1 - O(\delta/n)$ we can still find as many as good maps as in the two-

pass algorithm. The remaining analysis of the algorithm follows exactly as in the two pass algorithm. The memory used in the one-pass modification can be obtained by replacing one of the $1/\epsilon^2$ factor in Equation (3.7.2) with $1/\epsilon'^2$ and multiply with $\log \frac{n^2}{\delta}$,

$$O\left(\frac{\log^8 n}{\beta\epsilon^2}\frac{\log^2(1/\delta)}{\log^5 \alpha}\log^2 \frac{n^2}{\delta}\right)$$

bits. Thus we yield,

Theorem 3.7.10. There is a streaming one pass algorithm `Level1`, that given input stream $\mathcal{S}$ with frequency vector v, level base $\alpha = 1+\gamma$, importance $\beta > 0$, precision $\epsilon > 0$ and error probability δ, output a list $(\alpha' = 1+\Theta(\gamma), \widehat{b}_1, \widehat{b}_2, \ldots, \widehat{b}_t)$, where $t = \log n/\log \alpha'$, such that,

- for all $i \in [t]$, $\widehat{b}_i \leq b_i$;
- if i is a β-important level, then $\widehat{b}_i \geq (1-\epsilon)b_i$,

with probability at least $1-\delta$, using space $O\left(\frac{\log^8 n}{\beta\epsilon^2}\frac{\log^2(1/\delta)}{\log^5 \alpha}\log^2 \frac{n^2}{\delta}\right)$ bits.

3.8 Concluding Remarks of Chapter 3

There is obviously a $\text{poly}(\frac{1}{\epsilon}\log n)$ gap between our upper and lower bounds. For the l_p norms, $p > 2$, our lower bound is $\Omega(n^{1-2/p})$, matching the true space complexity to within a $\Theta(\log n)$ factor.[68,71] Despite the gap, we do partially answer Open Problem 30 (Universal Sketching) in,[75] by showing that the class of symmetric norms admits universal sketches, and also Open Problem 5 (Sketchable Distances) in,[75] by showing that every symmetric norm l

admits streaming algorithms and is thus sketchable with space $\mathrm{mmc}(l)^2 \cdot \mathrm{poly}(\frac{1}{\epsilon}\log n)$.

Both our algorithm and our lower bound rely on the symmetry of the norm. It would be very interesting to see whether the modulus of concentration is a key factor in the space complexity also for general norms. Our results do extend a little towards more general norms. Notice that, given any symmetric norm l on $\mathbb{R}^n$ and invertible linear transformation $A : \mathbb{R}^n \to \mathbb{R}^n$, our results also apply to the streaming complexity of $l_A := l(A(\cdot))$, which is always a norm but is generally not symmetric. For example, $l_2(A(\cdot))$ is the norm induced by the inner product $\langle x, y\rangle_A := y^T A^T A x$, and it is not symmetric unless all singular values of A are the same. To compute $l_A(v)$ one applies A to the incoming stream vector v and then runs an algorithm for l (we do not count the storage for A). Therefore, the space complexity of l_A is no worse than that of l, and, as the same argument applies to $l = l_A(A^{-1}(\cdot))$, the two must have the same streaming complexity (we assume that $O(\log n)$ bits suffice to represent any entry of A or A^{-1} to sufficient precision). More generally, norms that can be related to each other by composition with an invertible linear transformation, as above, must have the same space complexity. On the other hand, this operation does not preserve $\mathrm{mc}(l)$ or $\mathrm{mmc}(l)$. Perhaps a norm should be put into a "canonical form" that is more amenable to determining its space complexity. For example, the distorted Euclidean norm $l(v) = (v^T A^T A v)^{1/2}$, mentioned above, may have $\mathrm{mmc}(l)$ on the order of $\min\{\sqrt{n}, \sigma_1(A)/\sigma_n(A)\}$, but it can be seen immediately to have space complexity $\mathrm{poly}(\frac{1}{\epsilon}\log n) \cdot \mathrm{mmc}(l_2)^2$ bits (in fact $O(\frac{1}{\epsilon^2}\log n)$ bits), from the AMS algorithm[14] and by recognizing $l(v) = l_2(Av)$ (again assuming $O(\log n)$ bits represents any entry of A to sufficient precision). Can we use $\mathrm{mmc}(\cdot)$ to determine the space complexity of every norm?

It would be very interesting also to design a small sketch that is oblivious to the linear transformation. For instance, let $\mathcal{M}$ be a family of linear transformations where $l_A \neq l_B$ for all $A, B \in \mathcal{M}$. Is there a linear sketch that approximates the norm $l_A(v)$ for any streamed vector $v \in \mathbb{R}^n$ and linear transformation $A \in \mathcal{M}$? Observe that no small sketch can be oblivious to all linear transformations, since that would allow recovery of every coordinate of v.

Our Theorem 3.1.3 shows a quadratic space-approximation tradeoff for every symmetric norm. Previously, this was only known for the l_∞ norm due to Saks and Sun.[94] Investigating space-approxmation tradeoff is an interesting direction because such tradeoffs appear in the sketching lower bounds of,[60] however no matching algorithms are known for other specific norms of interests, such as the Earth Mover Distance and the trace norm (of matrices).

Chapter 4

k-median Clustering in Dynamic Streams

This chapter is based on BFHSY (2017).[100]

4.1 Background

The analysis of very large data sets is still a big challenge. Particularly, when we would like to obtain information from data sets that occur in the form of a data stream like, for example, streams of updates to a data base system, internet traffic and measurements of scientific experiments in astro- or particle physics (e.g.[8]). In such scenarios it is difficult and sometimes even impossible to store the data. Therefore, we need algorithms that process the data sequentially and maintain a summary of the data using space much smaller than

the size of the stream. Such algorithms are often called streaming algorithms (for more introduction on streaming algorithms, please refer to[7]).

One fundamental technique in data analysis is clustering. The idea is to group data into clusters such that data inside the same cluster is similar and data in different clusters is different. Center based clustering algorithms also provide for each cluster a cluster center, which may act as a representative of the cluster. Often data is represented as vectors in $\mathbb{R}^d$ and similarity between data points is often measured by the Euclidean distance. Clustering has many applications ranging from data compression to unsupervised learning.

In this paper we are interested in clustering problems over dynamic data streams, i.e. data streams that consist of updates, for example, to a database. Our stream consists of insert and delete operations of points from $\{1, \ldots, \Delta\}^d$. We assume that the stream is consistent, i.e. there are no deletions of points that are not in the point set and no insertions of points that are already in the point set. We consider the k-median clustering problem, which for a given a set of points $P \subseteq \mathbb{R}^d$ asks to compute a set C of k points that minimizes the sum of distances of the input points to their nearest points in C.

4.1.1 Our Results

We develop the first $(1+\epsilon)$-approximation algorithm for the k-median clustering problem in dynamic data streams that uses space polynomial in the dimension of the data. To our best knowledge, all previous algorithms required space exponentially in the dimension. Formally, our main theorem states,

Theorem 4.1.1 (Main Theorem)**.** Fix $\epsilon \in (0, 1/2)$, positive integers k and Δ, Algorithm 6 makes a single pass over the dynamic streaming point set $P \subset [\Delta]^d$, outputs a weighted set S, such that with probability at least 0.99, S is an ϵ-coreset for k-median of size $O\left(kd^4L^4/\epsilon^2\right)$, where $L = \log \Delta$. The algorithm uses $\widetilde{O}\left(kd^7L^7/\epsilon^2\right)$ bits in the worst case, processes each update in time $\widetilde{O}(dL^2)$ and outputs the coreset in time $\text{poly}(d, k, L, 1/\epsilon)$ after one pass of the stream.

The theorem is restated in Theorem 4.3.6 and the proof is presented in Section 4.3.3. The coreset we constructed may contain negatively weighted points. Thus naïve offline algorithms do not apply directly to finding k-clustering solutions on the coreset. We also provide an alternative approach that output only non-negatively weighted coreset. The new algorithm is slightly more complicated. The space complexity and coreset size is slightly worse than the one with negative weights but still polynomial in d, $1/\epsilon$ and $\log \Delta$ and optimal in k up to polylog$k$ factor.

Theorem 4.1.2 (Alternative Results)**.** Fix $\epsilon \in (0, 1/2)$, positive integers k and Δ, Algorithm 11 makes a single pass over the streaming point set $P \subset [\Delta]^d$, outputs a weighted set S with *non-negative weights* for each point, such that with probability at least 0.99, S is an ϵ-coreset for k-median of size $\widetilde{O}\left(kd^4L^4/\epsilon^2\right)$. The algorithm uses $\widetilde{O}\left(kd^8L^8/\epsilon^2\right)$bits in the worst case. For each update of the input, the algorithm needs $\text{poly}(d, 1/\epsilon, L, \log k)$ time to process and outputs the coreset in time $\text{poly}(d, k, L, 1/\epsilon)$ after one pass of the stream.

The theorem is restated in Theorem 4.4.3 in Section 4.4 and the proof is presented therein. Both approaches can be easily extended to maintain a coreset for a general metric space.

4.1.2 Our Techniques

From a high level, both algorithms can be viewed as a combination of the ideas introduced by Frahling and Sohler[50] with the coreset construction by Chen.[101]

To explain our high-level idea, we first summarize the idea of Chen.[101] In their construction, they first obtain a (α, β)-bi-criterion solution. Namely find a set of at most αk centers such that the k-median cost to these αk centers is at most $\beta\mathsf{OPT}$, where OPT is the optimal cost for a k-median solution. Around each of the αk points, they build logarithmically many concentric ring regions and sample points from these rings. Inside each ring, the distance from a point to its center is upper and lower bounded. Thus the contribution to the optimal cost from the points of this ring is lower bounded by the number of points times the inner diameter of the ring. To be more precise, their construction requires a partition of $\widetilde{O}(\alpha k)$ sets of the original data points satisfying the following property: for the partition $P_1, P_2, \ldots, P_{k'}$, $\sum_i |P_i|\mathrm{diam}(P_i) \lesssim \beta\mathsf{OPT}$. They then sample a set of points from each part to estimate the cost of an arbitrary k-set from $[\Delta]^d$ up to $O(\epsilon|P_i|\mathrm{diam}(P_i)/\beta)$ additive error. Combining the samples of the k' parts, this gives an additive error of at most $\epsilon\mathsf{OPT}$ and therefore an ϵ-coreset.

The first difficulty in generalizing the construction of Chen to dynamic streams is that it depends on first computing approximate centers, which seems at first glance to require two passes. Surprisingly (since we would like to be polynomial in d), we can resolve this difficulty using a grid-based construction. The grid structure can be viewed as a (2^d)-ary tree of cells. The root level of the tree is a single cell containing the entire set of points.

Going down a level through the tree, each parent cell is split evenly into 2^d subcells. Thus in total there are $\log_2 \Delta$ grid levels. Each cell of the finest level contains at most a single point.

Without using any information of a pre-computed (α, β)-bi-criterion solution to the k-median problem, as it does in,[101] our first idea (similar to the idea used in[49]) is to use a randomly shifted grid (i.e. shift each coordinate of the cell of the root level by a random value r, where r is uniformly chosen from $\{1, 2, \ldots \Delta\}$, and redefine the tree by splitting cells into 2^d subcells recursively). We show that with high probability, in each level, at most $\widetilde{O}(k)$ cells are close to (or containing) a center of an optimal solution to the k-median. For the remaining cells, we show that each of them cannot contain too many points, since otherwise they would contribute too much to the cost of the optimal solution (since each point in these cells is far away from each of the optimal centers). We call the cells containing too many points in a level *heavy* cells. The immediate non-heavy children of the heavy cells form a partition of the entire point sets (i.e. the cells that are not heavy, but have heavy parents). Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{k'}$ be these cells, and we can immediately show that $\sum_i |\mathcal{C}_i| \mathrm{diam}(\mathcal{C}_i) \leq \beta \mathsf{OPT}$ for some $\beta = O(d^{3/2})$. If we can identify the heavy cells (e.g. use heavy hitter algorithms), and sample points from their immediate non-heavy children in a dynamic stream, we will obtain a construction similar to Chen.[101]

Our second idea allows us to significantly reduce space requirements and also allows us to do the sampling more easily. For each point p, the cells containing it form a path on the grid tree. We write each point as a telescope sum as the cell centers on the path of the point (recall that the grids of each level are nested and the c^0 is the root of the tree). For example,

let $c^0, c^1, \ldots, c^L$ be the cell centers of the path, where $c_L = p$, and define $\mathbf{0}$ to be the zero vector. Then $p = c^L - c^{L-1} + c^{L-1} - c^{L-2} \ldots + c^0 - \mathbf{0} + \mathbf{0}$. In this way, we can represent the distance from a point to a set of points as the distance of cell centers to that set of points. For example, let $Z \subset [\Delta]^d$ be a set of points, and $d(p, Z)$ be the distance from p to the closest point in Z. Then $d(p, Z) = d(c^L, Z) - d(c^{L-1}, Z) + d(c^{L-1}, Z) - d(c^{L-2}, Z) + \ldots + d(\mathbf{0}, Z)$. Thus we can decompose the cost a set Z into $L + 2$ levels: the cost in level $l \in [0, L]$ is $\sum_p d(c_p^l, Z) - d(c_p^{l-1}, Z)$, where c_l^p is the center of the cell containing p in level l and the cost in the (-1)-st level is $|P|d(\mathbf{0}, Z)$, where P is the entire points set. Since $|d(c_p^l, Z) - d(c_p^{l-1}, Z)|$ is bounded by the cell diameter of the level, we can sample points from the non-heavy cells of the entire level, and guarantee that the cost of that level is well-approximated. Notice that (a) we do not need to sample $\widetilde{O}(k)$ points from every part of the partition, thus we save a k factor on the space and (b) we do not need to sample the actual points, but only an estimation of the number of points in each cell, thus the sampling becomes much easier (there is no need to store the sampled points).

In the above construction, we are able to obtain a coreset, but the weights can be negative due the the telescope sum. It is not easy find an offline k-median algorithm to output the solutions from a negatively-weighted coreset. To remove the negative weights, we need to adjust the weights of cells. But the cells with a small number of points (compared to the heavy cells) are problematic – the sampling-based empirical estimations of the number of points in them has too much error to be adjusted.

In our second construction, we are able to remove all the negative weights. The major difference is that we introduce a cut-off on the telescope sum. For example, $d(p, Z) =$

$d(c_p^L, Z) - d(c_p^{l(p)}, Z) + d(c_p^{l(p)}) - d(c_p^{l(p)-1}) + \ldots + d(\mathbf{0}, Z)$ where $l(p)$ is a cutoff level of point p such that the cell containing p in level $l(p)$ is heavy but no longer heavy in level $l(p)+1$. We then sample point p with some probability defined according to $l(p)$. In other words, we only sample points from heavy cells and not from non-heavy ones. Since a heavy cell contains enough points, the sampling-based estimation of the number of points is accurate enough and thus allows us to adjust them to be all positive.

Finally, to handle the insertion and deletions, we use a F_2-heavy hitter algorithm to identify the heavy cells. We use pseudo-random hash functions (e.g. Nisan's construction[16,99] or k-wise independent hash functions) to do the sampling and use a `K-Set` data structure[102] to store the sampled points in the dynamic stream.

4.1.3 Related Work

There is a rich history in studies of geometric problems in streaming model. Among these problems some excellent examples are: approximating the diameter of a point set,[103,104] approximately maitain the convex hull,[105,106] the min-volume bounding box,[107,108] maintain ϵ-nets and ϵ-approximations of a data stream.[109] Clustering problem is another interesting and popular geometric problem studied in streaming model. There has been a lot of works on clustering data streams for the k-median and k-means problem based on coresets.[28,29,101,110–112] Additionally[24,113,114] studied the problem in the more general metric space. The currently best known algorithm for k-median problem in this setting is an $O(1)$-approximation using $O(k\text{polylog}n)$ space.[105] However, all of the above methods do not work for dynamic streams.

The most relevant works to ours are those by Indyk,[49] Indyk & Price[115] and Frahling & Sohler.[50] Indyk[49] introduced the model for dynamic geometric data streamings. He studied algorithms for (the weight of) minimum weighted matching, minimum bichromatic matching and minimum spanning tree and k-median clustering. He gave a exhaustive search $(1+\epsilon)$ approximation algorithm for k-median and a (α, β)-bi-criterion approximation algorithm. Indyk & Price[115] studied the problem of sparse recovery under Earth Mover Distance. They show a novel connection between EMD/EMD sparse recovery problem to k-median clustering problem on a two dimensional grid. The most related work to current one is Frahling & Sohler,[50] who develop a streaming $(1+\epsilon)$-approximation algorithms for k-median as well as other problems over dynamic geometric data streams. All previous constructions for higher dimensional grid require space exponential in the dimension d.

4.2 Preliminaries

For integer $a \leq b$, we denote $[a] := \{1, 2, \ldots, a\}$ and $[a, b] := \{a, a+1, \ldots, b\}$ for integer intervals. We will consider a point set P from the Euclidean space $\{1, \ldots, \Delta\}^d$. Without loss of generality, we always assume Δ is of the form 2^L for some integer L, since otherwise we can always pad Δ without loss of a factor more than 2. Our streaming algorithm will process insertions and deletions of points from this space. We study the k-median problem, which is to minimize $\text{cost}(P, Z) = \sum_{p \in P} d(p, Z)$ among all sets Z of k centers from $\mathbb{R}^d$ and where $d(p, q)$ denotes the Euclidean distance between p and q and $d(p, Z)$ for a set of points Z denotes the distance of p to the closest point in Z. The following definition is from.[28]

Definition 4.2.1. Let $P \subseteq [\Delta]^d$ be a point set. A small weighted set S is called an ϵ-coreset for the k-median problem, if for every set of k centers $Z \subset [\Delta]^d$ we have [1]

$$(1-\epsilon)\cdot \mathrm{cost}(P,Z) \leq \mathrm{cost}(S,Z) \leq (1+\epsilon)\cdot \mathrm{cost}(P,Z),$$

where $\mathrm{cost}(S,Z) := \sum_{s\in S} \mathrm{wt}(s)d(s,Z)$ and $\mathrm{wt}(s)$ is the weight of point $s \in S$.

Through out the paper, we assume parameters $\epsilon, \rho, \delta, \in (0, \frac{1}{2})$ unless otherwise specified. For our algorithms and constructions we define a nested grid with L levels, in the following manner.

Definition of grids Let $v = (v_1, \ldots, v_d)$ be a vector chosen uniformly at random from $[0, \Delta - 1]^d$. Partition the space $\{1, \ldots, \Delta\}^d$ into a regular Cartesian grid $\mathcal{G}_0$ with side-length Δ and translated so that a vertex of this grid falls on v. Each cell of this grid can be expressed as $[v_1 + n_1\Delta, v_1 + (n_1+1)\Delta) \times \ldots \times [v_d + n_d\Delta, v_d + (n_d+1)\Delta)$ for some $(n_1, \ldots, n_d) \in \mathbb{Z}^d$. For $i \geq 1$, define the regular grid $\mathcal{G}_i$ as the grid with side-length $\Delta/2^i$ aligned such that each cell of $\mathcal{G}_{i-1}$ contains 2^d cells of $\mathcal{G}_i$. The finest grid is $\mathcal{G}_L$ where $L = \lceil \log_2 \Delta \rceil$; the cells of this grid therefore have side-length at most 1 and thus contain at most a single input point. Each grid forms a partition of the point-set $\mathcal{S}$. There is a d-ary tree such that each vertex at depth i corresponds to a cell in $\mathcal{G}_i$, and this vertex has 2^d children which are the cells of $\mathcal{G}_{i+1}$ that it contains. For convenience, we define $\mathcal{G}_{-1}$ as the entire dataset and it contains a single cell $\mathcal{C}_{-1}$. For each cell $\mathcal{C}$, we also treat it as a subset of the input points (i.e. $\mathcal{C} \cap P$) if there is no confusion.

[1] For simplicity of the presentation, we define the coreset for all sets of k centers $Z \subset [\Delta]^d$, but it can be generalized to all sets of k centers $Z \subset \mathbb{R}^d$ with an additional polylog$(1/\epsilon)$ factor in the space. We discuss this point further in Section 4.9.

We denote $Z^* \subset [\Delta]^d$ as the optimal solution for k-median and $\mathtt{OPT}$ as the optimal cost for Z^*. The proof of the following lemma is delayed to Section 4.5.

Lemma 4.2.2. Fix a set $Z \subset [\Delta]^d$, then with probability at least $1 - \rho$, for every level $i \in [0, L]$, the number of cells that satisfy $d(\mathcal{C}, Z) \leq \Delta/(2^{i+1}d)$ is at most $e|Z|(L+1)/\rho$.

4.2.1 Outline

In Section 4.3, we introduce the coreset with negative weights. In Section 4.4, we introduce a modified construction with all positive weights. Section 4.9 comes with the final remarks.

4.3 Generally Weighted Coreset

In this section, we present our generally weighted coreset construction. In Section 4.3.1, we introduce the telescope sum representation of a point p and the coreset framework. In Section 4.3.2, we illustrate our coreset framework with an offline construction. In Section 4.3.3 we present an one pass streaming algorithm that implements our coreset framework.

4.3.1 The Telescope Sum and Coreset Framework

Our first technical idea is to write each point as a telescope sum. We may interpret this sum as replacing a single point by a set of points in the following way. Each term $(p - q)$ of

the sum can be viewed as a pair of points p and q, where p has weight 1 and q has weight -1. The purpose of this construction is that the contribution of each term $(p-q)$ (or the corresponding two points) is bounded. This can be later exploited when we introduce and analyze our sampling procedure.

We now start to define the telescope sum, which will relate to our nested grids. For each $\mathcal{C} \in G_i$, denote $c(\mathcal{C})$ (or simply c) as its center. For each point $p \in P$, define $\mathcal{C}(p,i)$ as the cell that contains p in $\mathcal{G}_i$, and c_p^i is the center of $\mathcal{C}(p,i)$. Then we can write

$$p = c_p^{-1} + \sum_{i=0}^{L} c_p^i - c_p^{i-1}.$$

where we set $c_p^{-1} = \mathbf{0}$ (we also call this the cell center of the (-1)-st level for convenience). The purpose of this can be seen when we consider the distance of p to an arbitrary k-centers $Z \subset [\Delta]^d$, we can write the cost of a single point p, as

$$d(p,Z) = d(c_p^{-1}, Z) + \sum_{i=0}^{L} d(c_p^i, Z) - d(c_p^{i-1}, Z).$$

Note that $c_p^L = p$ since the cells of $\mathcal{G}_L$ contain a single point. Thus the cost of the entire set $\mathrm{cost}(P,Z)$ can be written as,

$$\sum_{i=0}^{L} \sum_{p \in P} d(c_p^i, Z) - d(c_p^{i-1}, Z) + \sum_{p \in P} d(c_p^{-1}, Z). \tag{4.3.1}$$

As one can see, we transform the cost defined using the original set of points to the "cost" defined using cell centers. To estimate the cost, it remains to estimate each of the terms,

$\sum_{p\in P} d(c_p^i, Z) - d(c_p^{i-1}, Z)$ for $i \in [0, L]$ and $\sum_{p\in P} d(c_p^{-1}, Z)$. In other words, assign weights to each of the centers of the grid cells. For $i \in [0, L]$, and a cell $\mathcal{C} \in \mathcal{G}_i$, denote $\mathcal{C}^P$ as the parent cell of $\mathcal{C}$ in grid $\mathcal{G}_{i-1}$. Thus we can rewrite the cost term as follows,

$$\begin{aligned}
\mathrm{cost}(\mathcal{G}_i, Z) &:= \sum_{p\in P} d(c_p^i, Z) - d(c_p^{i-1}, Z) \\
&= \sum_{\mathcal{C}\in\mathcal{G}_i} \sum_{p\in\mathcal{C}} d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \\
&= \sum_{\mathcal{C}\in\mathcal{G}_i} |\mathcal{C}| \left[d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z) \right] \\
&= \sum_{\mathcal{C}\in\mathcal{G}_i} |\mathcal{C}| d(c(\mathcal{C}), Z) - \sum_{\mathcal{C}'\in\mathcal{G}_{i-1}} \sum_{\mathcal{C}\in\mathcal{G}_i : \mathcal{C}\subset\mathcal{C}'} |\mathcal{C}| d(c(\mathcal{C}'), Z). \qquad (4.3.2)
\end{aligned}$$

For $i = -1$, we denote $\mathrm{cost}(\mathcal{G}_{-1}, Z) = |P| d(c_p^{-1}, Z)$. Then this leads to our following coreset construction framework.

Generally Weighted Construction The coreset S in the construction is composed by a weighted subset of centers of grid cells. The procedure of the construction is to assign some (integer) value to each cell center. For instance, maintain a integer valued function $\widehat{|\cdot|}$ on cells (using small amount of space). $\widehat{|\mathcal{C}|}$ is called the *value* of the cell $\mathcal{C}$. Let c be the center of $\mathcal{C}$, then the weight for c is

$$\mathrm{wt}(c) = \widehat{|\mathcal{C}|} - \sum_{\mathcal{C}' : \mathcal{C}'\in\mathcal{G}_{i+1}, \mathcal{C}'\subset\mathcal{C}} \widehat{|\mathcal{C}'|}. \qquad (4.3.3)$$

And for the L-th grid $\mathcal{G}_L$, the weight for each cell $\mathcal{C}$ is just $\widehat{|\mathcal{C}|}$. Note that there might be negative weights for some cells.

As a naïve example, we set $\widehat{|\mathcal{C}|} := |\mathcal{C}|$ as the exact number of points of a cell $\mathcal{C}$. Then we would expect the cells in every level except those in $\mathcal{G}_L$ have weight 0. In other words, we stored the entire point set as the coreset. As we will show, if we allow $\widehat{|C|}$ as an approximation of $|C|$ up to additive error, we can compress the number of non-zero weighted centers to be a smaller number.

Definition 4.3.1. Given a grid structure, and a real valued function $\widehat{|\cdot|}$ on the set of cells. We define a function $\widehat{\text{cost}} : [\Delta]^d \times \mathcal{G} \to \mathbb{R}$ as follows, for $i \in [0, L]$ and $Z \subset [\Delta]^d$,

$$\widehat{\text{cost}}(\mathcal{G}_i, Z) := \sum_{\mathcal{C} \in \mathcal{G}_i} \widehat{|\mathcal{C}|} d(c(\mathcal{C}), Z) - \sum_{\mathcal{C}' \in \mathcal{G}_{i-1}} \sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \subset \mathcal{C}'} \widehat{|\mathcal{C}|} \cdot d(c(\mathcal{C}'), Z), \tag{4.3.4}$$

and $\widehat{\text{cost}}(\mathcal{G}_{-1}, Z) = \widehat{|\mathcal{C}_{-1}|} d(\mathbf{0}, Z)$, where $\mathcal{C}_{-1}$ is the cell in $\mathcal{G}_{-1}$ containing the entire set of points.

Lemma 4.3.2. Fix an integer valued function $\widehat{|\cdot|}$ on the set of cells and parameter $0 < \epsilon < \frac{1}{2}$. Let S be the set of all cell centers with weights assigned by Equation (4.3.3). If $\widehat{|\mathcal{C}_{-1}|} = |P|$ (recall that $\mathcal{C}_{-1}$ is the first cell containing the entire dataset) and for any $Z \subset [\Delta]^d$ with $|Z| \leq k$ and $i \in [0, L]$

$$\left| \text{cost}(\mathcal{G}_i, Z) - \widehat{\text{cost}}(\mathcal{G}_i, Z) \right| \leq \frac{\epsilon \mathsf{OPT}}{L+1},$$

then S is an ϵ-coreset for k-median.

Proof. Given an arbitrary set of centers $Z \subset [\Delta]^d$,

$$\begin{aligned}
\mathrm{cost}(S, Z) &= \sum_{s \in S} \mathrm{wt}(s) d(s, Z) \\
&= \sum_{i \in [0,L]} \sum_{\mathcal{C} \in \mathcal{G}_i} d(c(\mathcal{C}), Z) \big(\widehat{|\mathcal{C}|} - \sum_{\substack{\mathcal{C}' : \mathcal{C}' \in \mathcal{G}_{i+1} \\ \mathcal{C}' \subset \mathcal{C}}} \widehat{|\mathcal{C}'|} \big) + |P| d(\mathbf{0}, Z) \\
&= \sum_{i \in [0,L]} \big[\sum_{\mathcal{C} \in \mathcal{G}_i} \widehat{|\mathcal{C}|} d(c(\mathcal{C}), Z) - \sum_{\substack{\mathcal{C}' \in \mathcal{G}_{i-1} \\ \mathcal{C} \in \mathcal{G}_i : \mathcal{C} \subset \mathcal{C}'}} \widehat{|\mathcal{C}|} d(c(\mathcal{C}'), Z) \big] + |P| d(\mathbf{0}, Z) \\
&= \sum_{i \in [0,L]} \widehat{\mathrm{cost}}(\mathcal{G}_i, Z).
\end{aligned}$$

It follows that $|\mathrm{cost}(S, Z) - \mathrm{cost}(P, Z)| \leq \epsilon \mathsf{OPT}$. □

4.3.2 An Offline Construction

In this section, we assume we have $(10, 10)$-bi-criterion approximation to k-median. Let $Z' = \{z'_1, z'_2, \ldots, z'_{10k}\}$ be the centers and o is the cost satisfying $\mathsf{OPT} \leq o \leq 10\mathsf{OPT}$. This can be done using.[16] We will show how we construct the coreset base on the framework described in the last section.

An Offline Construction For each point in level $\mathcal{G}_{-1}$, we sample it with probability $\pi_{-1} = 1$ (i.e. count the number of points exactly) and set $\widehat{|\mathcal{C}_{-1}|} := |P|$. For each level $i \in [0, L]$, we pick the set of all cells $\mathcal{C}$ satisfying $d(\mathcal{C}, Z') \leq W/(2d)$, where W is the side length of $\mathcal{C}$. Denote the set of these cells as $C_{Z'}$. We count the number of points in each of these cells exactly, and set $\widehat{|\mathcal{C}|} := |\mathcal{C}|$. For the points in the rest of cells, for each $i \in [0, L]$,

we sample the points with probability

$$\pi_i = \min\left(\frac{200(L+1)^2\Delta d^2}{2^i\epsilon^2 o}\ln\frac{2(L+1)\Delta^{kd}}{\rho}, 1\right) \tag{4.3.5}$$

uniformly and independently. Denote S_i as the set of sampled points at level i. For each $\mathcal{C} \notin C_{Z'}$, set

$$\widehat{|\mathcal{C}|} := |S_i \cap \mathcal{C}|/\pi_i.$$

Then, from the bottom level to the top level, we assign the weight to the cell centers of each of the cells and their parent cells with non-zero $\widehat{|\mathcal{C}|}$ using (4.3.3). Denote $\mathcal{S}$ as the coreset, which contains the set of cell centers of non-zero weight.

Theorem 4.3.3. Fix $\epsilon, \rho \in (0, 1/2)$, then with probability at least $1 - 8\rho$, the offline construction $\mathcal{S}$ is an ϵ-coreset for k-median and that

$$|\mathcal{S}| = O\left(\frac{d^4kL^4}{\epsilon^2}\log\frac{1}{\rho} + \frac{kL^2}{\rho}\right).$$

Proof of Theorem 4.3.3. By definition $\widehat{|\mathcal{C}_{-1}|} = |P|$, it is suffice to show that with probability at least $1 - 4\rho$, for every $i \in [0, L]$ and every k-set $Z \subset [\Delta]^d$,

$$\left|\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) - \mathrm{cost}(\mathcal{G}_i, Z)\right| \le \epsilon\mathsf{OPT}/(L+1).$$

It follows from Lemma 4.3.2 that, $\mathcal{S}$ is an ϵ-coreset.

Let S_i be the sampled points of level i. Fix a k-set $Z \subset [\Delta]^d$, for each $i \in [0, L]$, by equation (4.3.2), we have that, $\widehat{\mathrm{cost}}(\mathcal{G}_i, Z) = \sum_{\mathcal{C}\in C_{Z'}} \widehat{|\mathcal{C}|}\left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\right) +$

$\sum_{p\in\mathcal{S}_i}(d(c_p^i,Z)-d(c_p^{i-1},Z)))/\pi_i$. Note that $E(\widehat{\text{cost}}(\mathcal{G}_i,Z))=\text{cost}(\mathcal{G}_i,Z)$. The first term contributes 0 to the difference $\widehat{\text{cost}}(\mathcal{G}_i,Z)-\text{cost}(\mathcal{G}_i,Z)$ since each $\widehat{|\mathcal{C}|}$ is exact. It remains to bound the error contribution from the second part. Denote $A_2=\sum_{p\in\mathcal{S}_i}(d(c_p^i,Z)-d(c_p^{i-1},Z)))/\pi_i$. Recall that Z' is the centers of the bi-criterion solution and $C_{Z'}$ is the set of cells with distance less than $W/(2d)$ to Z', where W is the side-length of a cell. Let $\mathcal{A}$ be event that $|C_{Z'}|\le e|Z'|(L+1)^2/\rho=O(kL^2/\rho)$. By Lemma 4.2.2, $\mathcal{A}$ happens with probability at least $1-\rho$. Conditioning on $\mathcal{A}$ happening, for each point $p\in\mathcal{C}\notin C_{Z'}$, we have that $d(p,Z')\ge\text{diam}(\mathcal{C})/(2d^{3/2})$. Therefore, $\sum_{p\in\mathcal{C}\notin C_Z}\text{diam}(\mathcal{C})\le(2d^{3/2})\sum_{p\in\mathcal{C}\notin C_Z}d(p,Z')\le 20d^{3/2}\mathsf{OPT}$. By Lemma 4.3.4, with probability at least $1-\frac{\rho}{(L+1)\Delta^{kd}}$, $|A_2-E(A_2)|\le\frac{\epsilon\mathsf{OPT}}{L+1}$. Since there are at most Δ^{kd} many different k-sets from $[\Delta]^d$, thus, for a fixed $i\in[0,L]$ with probability at least $1-\frac{\rho}{L+1}$, for all k-sets $Z\subset[\Delta]^d$, $\left|\widehat{\text{cost}}(\mathcal{G}_i,Z)-\text{cost}(\mathcal{G}_i,Z)\right|\le\epsilon\mathsf{OPT}/(L+1)$. By the union bound, with probability at least $1-4\rho$, $\mathcal{S}$ is the desired coreset.

It remains to bound the size of $\mathcal{S}$. Conditioning on $\mathcal{A}$ happening, then $|C_{Z'}|=O(kL^2/\rho)$. For each level i, since each point from cells $\mathcal{C}\notin C_{Z'}$ contributes at least $\Delta/(2^{i+1}d)$ to the bi-criterion solution, there are at most $O(2^i\mathsf{OPT}d/\Delta)$ points in cells not in $C_{Z'}$. By a Chernoff bound, with probability at least $1-\rho/(L+1)$, the number of points sampled from cells $\mathcal{C}\notin C_{Z'}$ of level i is upper bounded by $O(d^4kL^3\log\frac{1}{\rho}/\epsilon^2)$. Thus for all levels, with probability at least $1-\rho$, the number of points sampled is upper bounded by $O(d^4kL^4\log\frac{1}{\rho}/\epsilon^2)$, which is also an upper bound of the number of cells occupied by sampled points. Now we bound the number of non-zero weighted centers. In the coreset construction, if a cell center has non-zero weight, then either itself or one of its children cells has non-zero assigned value $\widehat{|\mathcal{C}|}$. Thus the number of non-zero weigted centers is upper bound by 2 times the number of

non-zero valued cells. Thus $|\mathcal{S}| = O(d^4kL^4 \log \frac{1}{\rho}/\epsilon^2 + \frac{kL^2}{\rho})$. □

Lemma 4.3.4. Fix $\epsilon, \rho \in (0, 1/2)$, if a set of cells C from grid $\mathcal{G}_i$ satisfies $\sum_{\mathcal{C}\in C} |\mathcal{C}|\text{diam}(\mathcal{C}) \leq \beta\mathsf{OPT}$ for some $\beta \geq 2\epsilon/(3(L+1))$, let S be a set of independent samples from the point set $\cup\{\mathcal{C} \in C\}$ with probability

$$\pi_i \geq \min\left(\frac{3a(L+1)^2\Delta\sqrt{d}\beta}{2^i\epsilon^2 o} \ln \frac{2\Delta^{kd}(L+1)}{\rho}, 1\right)$$

where $0 < o \leq a\mathsf{OPT}$ for some $a > 0$, then for a fixed set $Z \subset [\Delta]^d$, with probability at least $1 - \rho/((L+1)\Delta^{kd})$,

$$\Big|\sum_{p\in S}(d(c_p^i, Z) - d(c_p^{i-1}, Z))/\pi_i - \sum_{p\in\cup\{\mathcal{C}\in C\}}(d(c_p^i, Z) - d(c_p^{i-1}, Z))\Big| \leq \frac{\epsilon\mathsf{OPT}}{L+1}.$$

The proof is a straightforward application of Bernstein inequality. It is presented in Section 4.5.

4.3.3 The Streaming Algorithm

For the streaming algorithm, the first challenge is that we do not know the actual value of OPT, neither do we have an (α, β)-bi-criterion solution. To handle this, we will show that we do not need an actual set of centers of an approximate solution, and that a conceptual optimal solution suffices. We will guess logarithmically many values for OPT to do the sampling. We re-run the algorithm in parallel for each guess of OPT.

The second challenge is that we cannot guarantee the sum $\sum_{\mathcal{C}\in\mathcal{G}_i}\text{diam}(\mathcal{C})$ to be upper

bounded by $\beta\texttt{OPT}$ as required in Lemma 4.3.4. We will show that we can split the set of cells into two parts. The first part satisfies the property that $\sum_{\mathcal{C}} |\mathcal{C}|\text{diam}(\mathcal{C}) \leq \beta\texttt{OPT}$ for some parameter β. The second part satisfies that $|\mathcal{C}|\text{diam}(\mathcal{C}) \geq a\texttt{OPT}/k$ for some constant a.

For the first part, we use a similar sampling procedure as we did in the offline case. The challenge here is that there might be too many points sampled when the algorithm is midway through the stream, and these points may be deleted later in the stream. To handle this case, we use a data structure called `K-Set` structure with parameter k.[102] We will insert (with deletions) a multiset of points $M \subset [N]$ into the `K-Set`. The data structure processes each stream operation in $O(\log(k/\delta))$ time. At each point of time, it supports an operation `RETRSET`, that with probability at least $1 - \delta$ either returns the set of items of M or returns `Fail`. Further, if the number of distinct items $|M|$ is at most k, then `RETRSET` returns M with probability at least $1 - \delta$. The space used by the `K-Set` data structure is $O(k(\log |M| + \log N) \log(k/\delta)$. The `K-Set` construction also returns the frequency of each stored points upon the `RETRSET` operation.

For the second part, we call these cells *heavy*. We first upper bound the number of heavy cells by αk for some $\alpha > 1$. We use a heavy hitter algorithm `HEAVY-HITTER` to retrieve an approximation to the number of points in these cells. The guarantee is given in the following theorem. In an insertion-deletion stream, it may that although the stream has arbitrary large length, at any moment a much smaller number of elements are active (that is, inserted and not yet deleted). We define the size of a stream to be the maximum number of active elements at any point of the stream.

Theorem 4.3.5 ([116] Theorem 2)**.** Fix $\epsilon, \delta \in (0, 1/2)$. Given a stream (of insertions and deletions) of size m consisting of items from universe $[n]$, there exists an algorithm HEAVY-HITTER(n, k, ϵ, δ) that makes a single pass over the stream and outputs a set of pairs H. With probability at least $1 - \delta$, the following holds,

(1) for each $(i, \widehat{f_i}) \in H$, $f_i^2 \geq \sum_{j=1}^{n} f_j^2/k - \epsilon^2 \sum_{j=k+1}^{n} f_j^2$;

(2) if for any $i \in [n]$ and $f_i^2 \geq \sum_{j=1}^{n} f_j^2/k + \epsilon^2 \sum_{j=k+1}^{n} f_j^2$, then $(i, \widehat{f_i}) \in H$;

(3) for each $(i, \widehat{f_i}) \in H$, $|\widehat{f_i} - f_i| \leq \epsilon \sqrt{\sum_{j=k+1}^{n} f_j^2}$.

The algorithm uses $O\left((k + \frac{1}{\epsilon^2}) \log \frac{n}{\delta} \log m\right)$ bits of space, $O(\log n)$ update time and $O(k + 1/\epsilon^2)$polylog(n) query time.

Thus, using HEAVY-HITTER, we are guaranteed that the error of the number of points in heavy cells is upper bounded by ϵ times the number of points in the non-heavy cells. The first heavy hitter algorithm that achieves an l_2 guarantee is by,[67] who has the same space and update time as that of the above algorithm. However the update time is slow, i.e. $O(n \log n)$ time to output the set of heavy hitters.

Lastly, we will use fully independent random hash function to sample the points. We will use Nissan's pseudorandom generator to de-randomize the hash functions by the method of.[16] Our main theorem for this section is as follows. The formal proof of this theorem is postponed to Section 4.6.

Theorem 4.3.6 (Main Theorem)**.** Fix $\epsilon, \rho \in (0, 1/2)$, positive integers k and Δ, Algorithm 6 makes a single pass over the streaming point set $P \subset [\Delta]^d$, outputs a weighted set S, such that with probability at least $1 - \rho$, S is an ϵ-coreset for k-median of size

$O\left(\frac{d^4L^4k}{\epsilon^2}+\frac{L^2k}{\rho}\right)$, where $L=\log\Delta$. The algorithm uses

$$O\left[k\left(\left(\frac{d^7L^7}{\epsilon^2}+\frac{d^3L^5}{\rho}\right)\log\frac{dkL}{\rho\epsilon}+\frac{d^5L^6}{\epsilon^2\rho}\right)\right]$$

bits in the worst case, processes each update in time $O\left(dL^2\log\frac{dkL}{\rho\epsilon}\right)$ and outputs the coreset in time $\text{poly}(d,k,L,1/\epsilon)$ after one pass of the stream.

4.4 Positively Weighted Coreset

In this section, we will introduce a modification to our previous coreset construction, which leads to a coreset with all positively weighted points. The full algorithm and proofs are postponed to Section 4.7. We present the main steps in this section.

The high level idea is as follows. When considering the estimate of the number of points in a cell, the estimate is only accurate when it truly contains a large number of points. However, in the construction of the previous section, we sample from each cell of each level, even though some of the cells contain a single point. For those cells, we cannot adjust their weights from negative to positive, since doing so would introduce large error. In this section, we introduce an ending level to each point. In other words, the number of points of a cell is estimated by sampling only if it contains many points. Thus, the estimates will be accurate enough and allow us to rectify the weights to be all positive.

4.4.1 Reformulation of the Telescope Sum

Definition 4.4.1. A *heavy cell identification scheme* $\mathcal{H}$ is a map $\mathcal{H} : \mathcal{G} \to \{\text{heavy, non-heavy}\}$ such that, $h(\mathcal{C}_{-1})$ =heavy and for cell $\mathcal{C} \in \mathcal{G}_i$ for $i \in [0, L]$

1. if $|\mathcal{C}| \geq \frac{2^i \rho d \mathsf{OPT}}{k(L+1)\Delta}$ then $\mathcal{H}(\mathcal{C}) = \text{heavy}$;

2. If $\mathcal{H}(\mathcal{C}) = \text{non-heavy}$, then $\mathcal{H}(\mathcal{C}') = \text{non-heavy}$ for every subsell $\mathcal{C}'$ of $\mathcal{C}$.

3. For every cell $\mathcal{C}$ in level L, $\mathcal{H}(\mathcal{C}) = \text{non-heavy}$.

4. For each $i \in [0, L]$, $|\{\mathcal{C} \in \mathcal{G}_i : \mathcal{H}(\mathcal{C}) = \text{heavy}\}| \leq \frac{\lambda_1 kL}{\rho}$, where $\lambda_1 \leq 10$ is a positive universal constant.

The output for a cell not specified by the above conditions can be arbitrary. We call a cell *heavy* if it is identified heavy by $\mathcal{H}$. Note that a heavy cell does not necessarily contain a large number of points, but the total number of these cells is always bounded.

In the sequel, heavy cells are defined by an arbitrary fixed identification scheme unless otherwise specified.

Definition 4.4.2. Fix a heavy cell identification scheme $\mathcal{H}$. For level $i \in [-1, L]$, let $\mathcal{C}(p, i) \in \mathcal{G}_i$ be the cell in $\mathcal{G}_i$ containing p. The *ending level* $l(p)$ of a point $p \in P$ is the largest level i such that $\mathcal{H}(\mathcal{C}(p, i))$ =heavy, and $\mathcal{H}(\mathcal{C}(p, i+1))$ =non-heavy.

Note that the ending level is uniquely defined if a heavy cell identification scheme is fixed. We now rewrite the telescope sum for p as follows,

$$p = \sum_{i=0}^{l(p)} \left(c_p^i - c_p^{i-1}\right) + c_p^L - c_p^{l(p)},$$

where $c_p^{-1} = \mathbf{0}$ and $c_p^L = p$. For arbitrary k-centers $Z \subset [\Delta]^d$, we write, $d(p, Z) =$

$\sum_{i=0}^{l(p)} \left(d(c_p^i, Z) - d(c_p^{i-1}, Z)\right) + d(c_p^L, Z) - d(c_p^{l(p)}, Z) + d(\mathbf{0}, Z)$.

4.4.2 The New Construction (with arbitrary weights)

For these heavy cells, we use `HEAVY-HITTER` algorithms to obtain accurate estimates of the number of points in these cells, thus providing a *heavy cell identification scheme*. For the non-heavy cells, we only need to sample points from the bottom level, $\mathcal{G}_L$, but with a different probability for points with different ending levels.

We now describe the new construction. This essentially has the same gaurantee as the simpler construction from the previous section, however the benefit here is that (as shown in the next subsection) it can be modified to output only positive weights. In the following paragraph, the estimations $\widehat{|\mathcal{C}|}$ are given as a blackbox. In proposition 4.7.9 we specify the conditions these estimations must satisfy.

Non-Negatively Weighted Construction Fix an arbitrary heavy cell identification scheme $\mathcal{H}$. Let P_l be all the points with ending level $l(p) = l$. For each heavy cell $\mathcal{C}$, let $\widehat{|\mathcal{C}|}$ be an estimation of number of points of $|\mathcal{C}|$, we also call $\widehat{|\mathcal{C}|}$ the *value* of cell $\mathcal{C}$. For each non-heavy cell $\mathcal{C}'$, let $\widehat{|\mathcal{C}'|} = 0$. Let S be a set samples of P constructed as follows: $S = S_{-1} \cup S_0 \cup S_1, \cup \ldots \cup S_L$, where S_l is a set of i.i.d samples from P_l with probability π_l. Here π_l for $l \in [-1, L]$ is redefined as $\pi_l =$

$$\min\left(\frac{\lambda_3 d^2 \Delta L^2}{2^l \epsilon^2 o} \log\left(\frac{2L\Delta^{dk}}{\rho}\right) + \frac{\lambda_4 d^2 k L^3 \Delta}{2^i \epsilon^2 \rho o} \log\frac{30kL^2}{\rho^2}, 1\right)$$

where $\lambda_3 > 0$ and $\lambda_4 > 0$ are universal constants. Our coreset $\mathcal{S}$ is composed by all the sampled points in S and the cell centers of heavy cells, with each point p assigned a weight

$1/\pi_{l(p)}$ and for each cell center c of a heavy cell $\mathcal{C} \in \mathcal{G}_i$, the weight is,

$$\mathrm{wt}(c) = \widehat{|\mathcal{C}|} - \sum_{\substack{\mathcal{C}':\mathcal{C}' \in \mathcal{G}_{i+1}, \mathcal{C}' \subset \mathcal{C}, \\ \mathcal{C}' \text{ is heavy}}} \widehat{|\mathcal{C}'|} - \frac{|S_i \cap \mathcal{C}|}{\pi_i}. \tag{4.4.1}$$

For each non-heavy cell $\mathcal{C}$ except for those in the bottom level, $\mathrm{wt}(c(\mathcal{C})) = 0$. The weight of each point from S is the value of the corresponding cell in the bottom level.

4.4.3 Ensuring Non-Negative Weights

We now provide a procedure to rectify all the weights for the coreset constructed in the last sub-section. The idea is similar to the method used in.[115] The procedure is shown in Algorithm 8. After this procedure, there will be no negative weights in the coreset outputs.

Theorem 4.4.3. Fix $\epsilon, \rho \in (0, 1/2)$, positive integers k and Δ, Algorithm 11 makes a single pass over the streaming point set $P \subset [\Delta]^d$, outputs a weighted set S with *non-negative weights* for each point, such that with probability at least 0.99, S is an ϵ-coreset for k-median of size

$$O\left[\frac{d^3 L^4 k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\right]$$

where $L = \log \Delta$. The algorithm uses

$$O\left[\frac{d^7 L^7 k}{\epsilon^2}\left(\rho d L + \frac{L}{\rho}\log^2\frac{dkL}{\rho\epsilon}\right)\log^2\frac{dkL}{\rho\epsilon}\right]$$

bits in the worst case. For each update of the input, the algorithm needs poly $(d, 1/\epsilon, L, \log k)$ time to process and outputs the coreset in time poly$(d, k, L, 1/\epsilon, 1/\rho, \log k)$ after one pass

of the stream.

4.5 Proofs of Section 4.3

Proof of Lemma 4.2.2. Fix an i and consider a grid $\mathcal{G}_i$. For each center z_j, denote $X_{j,\alpha}$ the indicator random variable for the event that the distance to the boundary in dimension α of grid $\mathcal{G}_i$ is at most $\Delta/(2^{i+1}d)$. Since in each dimension, if the center is close to a boundary, it contributes a factor at most 2 to the total number of close cells. It follows that the number of cells that have distance at most $\Delta/(2^{i+1}d)$ to z_j is at most,

$$N = 2^{\sum_{\alpha=1}^{d} X_{j,\alpha}}.$$

Defining $Y_{j,\alpha} = 2^{X_{j,\alpha}}$, we obtain,

$$E[N] = E\left[\prod_{\alpha=1}^{d} Y_{j,\alpha}\right] = \prod_{\alpha=1}^{d} E[Y_{j,\alpha}].$$

We have that $Pr[X_{j,\alpha} = 1] \leq 1/d$ and so we get,

$$E[Y_{j,\alpha}] \leq E\left[1 + X_{j,\alpha}\right] = 1 + E[X_{j,\alpha}] \leq 1 + 1/d.$$

Thus $E(N) = \prod_{\alpha=1}^{d} E[Y_{j,\alpha}] \leq (1 + 1/d)^d \leq e$. Thus the expected number of center cells is at most $(1 + 1/d)^d|Z| \leq e|Z|$. By Markov's inequality, the probability that we have more than $e|Z|(L + 1)/\rho$ center cells in each grid is at most $\rho/(L + 1)$. By a union bound, the probability that in any grid we have more than $e|Z|(L + 1)/\rho$ center cells is at most ρ. □

Proof of Lemma 4.3.4. Let $L' = L + 1$. Note that for each point $p \in P$, $|d(c_p^i, Z) - d(c_p^{i-1}, Z)| \leq \Delta\sqrt{d}/2^i$. Denote $\widehat{A} = \sum_{p\in S}(d(c_p^i, Z) - d(c_p^{i+1}, Z))/\pi_i$ and $A = \sum_{p\in\cup\{\mathcal{C}\in C\}}(d(c_p^i, Z) - d(c_p^{i+1}, Z))$. We have that $E(\widehat{A}) = A$. Let

$$X_p := \mathbb{I}_{p\in S}(d(c_p^i, Z) - d(c_p^{i+1}, Z))/\pi_i,$$

where $\mathbb{I}_{p\in S}$ is the indicator function that $p \in S$. Then we have that $Var(X_p) \leq \Delta^2 d/(4^i\pi_i)$ and $b := \max_p |X_p| \leq \Delta\sqrt{d}/(2^i\pi_i)$. By Bernstein's inequality,

$$\begin{aligned} Pr\left[|\widehat{A} - A| > t\right] &\leq 2e^{-\frac{t^2}{2|P|\Delta^2 d/(4^i\pi_i)+2bt/3}} \\ &\leq 2e^{-\frac{3\times 2^{i-1}t^2\pi_i}{(\beta\mathsf{OPT}+\frac{t}{3})\Delta\sqrt{d}}}. \end{aligned} \tag{4.5.1}$$

By setting $t = \epsilon\mathsf{OPT}/L'$, we have that

$$Pr\left[|\widehat{A} - A| > \frac{\epsilon\mathsf{OPT}}{L'}\right] \leq 2e^{-\ln\frac{2L'\Delta^{dk}}{\rho}} \leq \frac{\rho}{L'\Delta^{dk}}. \tag{4.5.2}$$

Thus with probability $1 - \rho/(L'\Delta^{dk})$, $\widehat{A}$ is an $\epsilon\mathsf{OPT}/L'$ additive approximation to the sum $\sum_{p\in P} d(c_p^i, Z) - d(c_p^{i+1}, Z)$. □

4.6 Proof of Theorem 4.3.6

Before we prove this theorem, we first present Lemma 4.6.1 and Lemma 4.6.2. In Algorithm 6, for each level $i \in [0, L]$, let H_i be the set of cells in $\mathcal{G}_i$ whose frequencies are

returned by `HEAVY-HITTER` in the `RetrieveFrequency` procedure. For each $\mathcal{C} \in H_i$, let $\widehat{|\mathcal{C}|}$ be the returned frequency of $\mathcal{C}$. Let H_i' be the set of cells in $\mathcal{G}_i$ whose frequencies are returned by a `K-set` in the `RetrieveFrequency` procedure. Then H_i and H_i' are complements in $\mathcal{G}_i$.

Lemma 4.6.1. Let $L' = L + 1$. Fix $\epsilon, \rho \in (0, 1/2)$. Let $Z^* \subset [\Delta]^d$ be a set of optimal k-centers for the k-median problem of the input point set. For each $i \in [0, L]$, if at most ekL'/ρ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \leq \Delta/(2^{i+1}d)$, then with probability $1 - \rho/L'$, the following two statements hold:

1. $\left|\sum_{\mathcal{C}\in H_i}(\widehat{|\mathcal{C}|} - |\mathcal{C}|)\left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\right)\right| \leq \frac{\epsilon \mathsf{OPT}}{2L'}$ for every $Z \subset [\Delta]^d$.

2. $\sum_{\mathcal{C}\in H_i'} |\mathcal{C}| diam(\mathcal{C}) \leq \beta \mathsf{OPT}$ for $\beta = 3d^{3/2}$

Proof of Lemma 4.6.1. Let $L' = L + 1$. Fix a value $i \in [0, L]$ and then $W = \Delta/2^i$ is the width of a cell in $\mathcal{G}_i$. Since at most ekL'/ρ cells in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \leq W/(2d)$, then of the remaining cells, at most $2kL'/\rho$ cells can contain more than $\rho d\mathsf{OPT}/(WkL')$ points. This is because each such cells contribute at least $\frac{\rho d \mathsf{OPT}}{WkL'}\frac{W}{2d} = \frac{\rho \mathsf{OPT}}{2kL'}$ to the cost which sums to OPT. Therefore, at most $(e+2)L'k/\rho$ cells contain more than $\rho d\mathsf{OPT}/(WkL')$ points.

The number of cells in grid $\mathcal{G}_i$ is at most $N = (1 + 2^i)^d$ (and perhaps as few as 2^{id}, depending on the random vector v), so `HEAVY-HITTER` receives cells of at most N types. Enumerating all cells $C \in \mathcal{G}_i$ such that $|C_j| \geq |C_{j+1}|$, define $f_j = |C_j|$. Algorithm 1 sets $k' = (e+2)L'k/\rho$, and the additive error of the estimator of f_i of `HEAVY-HITTER` is given by $\epsilon'\sqrt{\sum_{j=k'+1}^{N} f_j^2}$. We know that for all $j > k'$ the value $f_j \leq \rho d\mathsf{OPT}/(WkL')$. Moreover, the sum $\sum_{j=k'+1}^{N} f_j \leq 2d\mathsf{OPT}/W$ because each point is at distance at least $W/(2d)$ to a point of Z^*. Under these two restraints, the grouping of maximal error is with $f_j = \rho d\mathsf{OPT}/(WkL')$

for $k' < j \leq k' + 2kL'/\rho$ and $f_j = 0$ for $j > k' + 2kL'/\rho$. Then the additive error becomes $\epsilon' \sqrt{2\rho/(kL')} d\mathsf{OPT}/W$.

The error from a single cell $\mathcal{C}_j$ is at most $|f_j - \widehat{f_j}|\sqrt{d}W$, and HEAVY-HITTER gaurantees with probability $1-\delta$ that $|f_j - \widehat{f_j}| \leq \epsilon' \sqrt{2\rho/(kL')} d\mathsf{OPT}/W$ for every j. Therefore to ensure total error over all k' cells is bounded by $\epsilon\mathsf{OPT}/(2L')$, we set $\epsilon' \leq \epsilon \sqrt{\frac{\rho}{8(2+e)^2 k d^3 L'^3}}$. Setting $\delta = \rho/L'$, the above bound holds with probability at least $1 - \rho/L'$.

For the second claim, we must bound $\sum_{\mathcal{C} \in H_i'} |\mathcal{C}|$. H_i consists of the top k' cells when ordered by value of $\widehat{f_j}$. This may differ from the top k' cells when ordered by value of f_j, but if j and j' change orders between these two orderings then $|f_j - f_{j'}| \leq 2\epsilon' \sqrt{2\rho/(kL')} d\mathsf{OPT}/W$. Since the sum may swap up to k' indices, the difference is bounded by $2k'\epsilon' \sqrt{2\rho/(kL')} d\mathsf{OPT}/W$. By setting $\epsilon' \leq \sqrt{\frac{\rho}{8(2+e)^2 dkL'}}$, we can ensure that the difference is at most $d\mathsf{OPT}/W$. We know that $\sum_{j=k'+1}^{N} f_j \leq 2d\mathsf{OPT}/W$, and so $\sum_{\mathcal{C} \in H_i'} |\mathcal{C}| \leq 3d\mathsf{OPT}/W$. For all cells $\mathcal{C} \in \mathcal{G}_i$, $\mathrm{diam}(\mathcal{C}) = \sqrt{d}W$. Therefore $\sum_{\mathcal{C} \in H_i'} |\mathcal{C}| \mathrm{diam}(\mathcal{C}) \leq 3d^{3/2}\mathsf{OPT}$.

□

Lemma 4.6.2. Let $L' = L + 1$. In Algorithm 6, fixing $\epsilon, \rho \in (0, 1/2)$, $o \in O$ and $i \in [0, L]$, if $\mathsf{OPT}/2 \leq o \leq \mathsf{OPT}$, then with probability $1 - \rho/(L'\Delta^{kd})$, at most $\frac{(2+e)L'k}{\rho} + \frac{24d^4 L'^3 k}{\epsilon^2} \ln \frac{1}{\rho}$ cells of $\mathcal{G}_i$ contain a point of $S_{i,o}$.

Proof. Similar to the proof of Lemma 4.6.1, there are at most $k' = (2+e)L'k/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ that satisfy $|\mathcal{C}| \geq \rho d\mathsf{OPT}/(Wk)$ and/or $d(\mathcal{C}, Z^*) \leq W/(2d)$. Considering the other cells, together they contain at most $2d\mathsf{OPT}/W$ points. So by a Chernoff bound, with probability $1 - \rho/(L'\Delta^{kd})$ at most $O(2d\pi_{i,o}\mathsf{OPT}/(W\rho)) \leq 24d^4 L'^3 k \ln \frac{1}{\rho}/\epsilon^2$ points are sampled. The

claim follows since each non-empty cell must contain at least one point. □

Proof of Theorem 4.3.6. Let $L' = L + 1$. W.l.o.g. we assume $\rho \geq \Delta^{-d}$, since otherwise we store the entire set of points and the theorem is proved. By Lemma 4.2.2, with probability at least $1 - \rho$, for every level $i \in [0, L]$, at most ekL'/ρ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(C, Z^*) \leq \Delta/(2^{i+1}d)$. Conditioning on this event, we will show 1) in the query phase, if $o^* \leq \mathtt{OPT}$, then with probability at least $1 - 4\rho$, S is the desired coreset; 2) there exists $o \leq \mathtt{OPT}$ in the guesses $O = \{1, 2, 4, \ldots, \Delta^{d+1}\}$ such that with probability $1 - 4\rho$, none of the K-set structures return `Nil`. 1) and 2) guarantee the correctness of the algorithm. Note that one can always rescale ρ to $\rho/9$ to achieve the correct probability bound. Finally, we will bound the space, update time and query time of the algorithm.

To show 1), we first note that the coreset size is at most $O(KL)$ as desired. Then by Lemma 4.3.2, we only need to show that with probability at least $1 - 4\rho$, for any k-set $Z \subset [\Delta]^d$ and any level $i \in [-1, L]$,

$$|\mathrm{cost}(\mathcal{G}_i, Z) - \widehat{\mathrm{cost}}(\mathcal{G}_i, Z)| \leq \frac{\epsilon \mathsf{OPT}}{L'},$$

where the value of each $\widehat{|C|}$ is returned by `RetrieveFrequency`. For each level i, we denote C_i as the set of cells that gets frequency from a `HEAVY-HITTER` instances in the `RetrieveFrequency` procedure, and $S_i = \{p \in \mathcal{C} : \mathcal{C} \notin C_i, h_{o^*,i}(p) = 1\}$ be the set of points sampled in the rest of cells. Since $KS_{o^*,i}$ does not return `Fail`, then for each $\mathcal{C} \in \mathcal{G}_i \backslash C_i$,

$\widehat{|\mathcal{C}|} = |S_i \cap \mathcal{C}|/\pi_i(o^*)$. Fix a k-set $Z \subset [\Delta]^d$, we rewrite the cost as,

$$\widehat{\text{cost}}(\mathcal{G}_i, Z) = \sum_{\mathcal{C} \in C_i} \widehat{|\mathcal{C}|} \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\right) + \sum_{p \in S_i} (d(c_p^i, Z) - d(c_p^{i-1}, Z)))/\pi_i(o^*),$$

where $\mathcal{C}^P$ is the parent cell of $\mathcal{C}$ in grid $\mathcal{G}_{i-1}$. By Lemma 4.6.1 we have that, with probability at least $1 - \rho/L'$, for every $Z \subset [\Delta]^d$,

$$\left| \sum_{\mathcal{C} \in C_i} \widehat{|\mathcal{C}|} \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\right) - \sum_{\mathcal{C} \in C_i} |\mathcal{C}| \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\right) \right| \leq \frac{\epsilon \mathsf{OPT}}{2L'},$$

and that, $\sum_{\mathcal{C} \in \mathcal{G}_i \setminus C_i} |\mathcal{C}| \text{diam}(\mathcal{C}) \leq 3d^{3/2}\mathsf{OPT}$. Conditioning on this event, by Lemma 4.3.4, with probability at least $1 - \rho/(L'\Delta^{kd})$,

$$\left| \sum_{p \in S_i} (d(c_p^i, Z) - d(c_p^{i-1}, Z)))/\pi_i - \sum_{\mathcal{C} \in \mathcal{G}_i \setminus C_i} |\mathcal{C}| \left(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)\right) \right| \leq \frac{\epsilon \mathsf{OPT}}{2L'}.$$

By a union bound, we show with probability at least $1 - 4\rho$, for any k-set $Z \subset [\Delta]^d$ and any level $i \in [-1, L]$,

$$|\text{cost}(\mathcal{G}_i, Z) - \widehat{\text{cost}}(\mathcal{G}_i, Z)| \leq \frac{\epsilon \mathsf{OPT}}{L'},$$

as desired.

To show 2), we will consider some $\mathsf{OPT}/2 \leq o \leq \mathsf{OPT}$. By Lemma 4.6.2 with probablity at least $1 - \rho/\Delta^{kd}$, the total number of cells occupied by sample points in each level is upper bounded by $K = \frac{(2+e)L'k}{\rho} + \frac{24d^4L'^3k}{\epsilon^2} \ln \frac{1}{\rho}$. Thus by the guarantee of the `K-Set` structure, with probability at least $1 - \rho$, none of the $\mathtt{KS}_{o,0}, \mathtt{KS}_{o,1} \ldots, \mathtt{KS}_{o,L}$ will return `Fail`.

The memory requirement of the algorithm is determined by the L instances of `HEAVY-HITTER` and the dL^2 instances of K-set. By Theorem 4.3.5, each instance of `HEAVY-HITTER` requires $O\left((k' + \frac{1}{\epsilon'^2}) \log \frac{N}{\delta} \log m\right)$ bits of space. Here $N \leq (1 + \Delta/W)^d \leq \Delta^d$ and m is the maximum number of elements active in the stream. Since we require that at most one point exists at each location at the same time, we have that $m \leq N$. The parameters are set to $k' = (2 + e)Lk/\rho$, $\epsilon' = \left(\epsilon \sqrt{\frac{\rho}{8(2+e)^2 k d^3 L^3}}\right)$, and $\delta = \rho/L$. This translates to a space bound of $O\left(dL + \log \frac{1}{\rho}\right) \frac{d^4 L^5 k}{\rho \epsilon^2}$ bits. For each `K-Set` data structure, it requires

$$O(KdL \log(KL/\rho)) = O\left(\frac{d^5 L^4 k}{\epsilon^2} + \frac{dkL^2}{\rho}\right) \log \frac{dkL}{\epsilon\rho}$$

bits of space. In total, there are $O(dL^2)$ `K-Set` instances and thus all `K-Set` instances cost $O\left(\frac{d^6 L^6 k}{\epsilon^2} + \frac{d^2 k L^4}{\rho}\right) \log \frac{dkL}{\epsilon\rho}$ bits of space. By the same argument as in the offline case, the last paragraph of the proof of Theorem 4.3.3, the size of the coreset is at most $O((k' + K)L) = O(d^4 k L^4 \epsilon^{-2} + kL^2/\rho)$ points. Finally, to derandomize the fully random functions, we use Nissan's pseudorandom generator[99] in a similar way used in[16]. But our pseudo-random bits only need to fool the sampling part of the algorithm rather than whole algorithm. We consider an augmented streaming algorithm $\mathcal{A}$ that does exactly the same as in `CoreSet` but with all the `HEAVY-HITTER` operations removed. Thus all K-set instances will have identical distribution with the ones in algorithm `CoreSet`. $\mathcal{A}$ uses $O(KdL \log(KL/\rho))$ bits of space. To fool this algorithm, using Nissan's pseudo-random generator, the length of random seed to generate the hash functions we need is of size $O(KdL \log(KL/\rho) \log(|O|\Delta^d)) = O\left(\left(\frac{d^7 k L^7}{\epsilon^2} + \frac{d^3 k L^5}{\rho}\right) \log \frac{dkL}{\rho\epsilon}\right)$. This random seed is thus sufficient to be used in Algorithm `CoreSet`. Thus the total space used in the algorithm is $O\left(\left(\frac{d^7 k L^7}{\epsilon^2} + \frac{d^3 k L^5}{\rho}\right) \log \frac{dkL}{\rho\epsilon} + \frac{d^5 k L^6}{\epsilon^2 \rho}\right)$

bits.

Regarding the update time, for the `HEAVY-HITTER` operations, it requires $O(L \log N) = O(dL^2)$ time. For the K-set operations, it requires $|O|LO(\log(KL/\rho)) = dL^2 \log(dkL/(\rho\epsilon))$ time. The de-randomized hash operation takes $O(dL)$ more time per update. The final query time is dominated by the `HEAVY-HITTER` data structure, which requires $\text{poly}(d, k, L, 1/\epsilon)$ time. □

4.7 Full Construction of Positively Weighted Coreset

In this section, we will introduce a modification to our previous coreset construction, which leads to a coreset with all positively weighted points. The high level idea is as follows. When considering the estimate of the number of points in a cell, the estimate is only accurate when it truly contains a large number of points. However, in the construction of the previous section, we sample from each cell of each level, even though some of the cells contain a single point. For those cells, we cannot adjust their weights from negative to positive, since doing so would introduce large error. In this section, we introduce an ending level to each point. In other words, the number of points of a cell is estimated by sampling only if it contains many points. Thus, the estimates will be accurate enough and allow us to rectify the weights to be all positive.

This section is organized as follows. We reformulate the telescope sum in Subsection 4.1, provide a different construction (still with negative weights) in Subsection 4.2, modify our different construction to output non-negative weights in Subsection 4.3, and move this

construction into to the streaming setting in Subsection 4.4. For simplicity of presentation, we will use $\lambda_1, \lambda_2, \ldots$ to denote some fixed positive universal constants.

4.7.1 Reformulation of the Telescope Sum

Definition 4.7.1. A *heavy cell identification scheme* $\mathcal{H}$ is a map $\mathcal{H} : \mathcal{G} \to \{\text{heavy, non-heavy}\}$ such that, $h(\mathcal{C}_{-1})$ =heavy and for cell $\mathcal{C} \in \mathcal{G}_i$ for $i \in [0, L]$

1. if $|\mathcal{C}| \geq \frac{2^i \rho d \mathsf{OPT}}{k(L+1)\Delta}$ then $\mathcal{H}(\mathcal{C}) = \text{heavy}$;
2. If $\mathcal{H}(\mathcal{C}) = \text{non-heavy}$, then $\mathcal{H}(\mathcal{C}') = \text{non-heavy}$ for every subsell $\mathcal{C}'$ of $\mathcal{C}$.
3. For every cell $\mathcal{C}$ in level L, $\mathcal{H}(\mathcal{C}) = \text{non-heavy}$.
4. For each $i \in [0, L]$, $|\{\mathcal{C} \in \mathcal{G}_i : \mathcal{H}(\mathcal{C}) = \text{heavy}\}| \leq \frac{\lambda_1 k L}{\rho}$, where $\lambda_1 \leq 10$ is a positive universal constant.

The output for a cell not specified by the above conditions can be arbitrary. We call a cell *heavy* if it is identified heavy by $\mathcal{H}$. Note that a heavy cell does not necessarily contain a large number of points, but the total number of these cells is always bounded.

In the sequel, heavy cells are defined by an arbitrary fixed identification scheme unless otherwise specified.

Definition 4.7.2. Fix a heavy cell identification scheme $\mathcal{H}$. For level $i \in [-1, L]$, let $\mathcal{C}(p, i) \in \mathcal{G}_i$ be the cell in $\mathcal{G}_i$ containing p. The *ending level* $l(p)$ of a point $p \in P$ is the largest level i such that $\mathcal{H}(\mathcal{C}(p, i))$ =heavy, and $\mathcal{H}(\mathcal{C}(p, i+1))$ =non-heavy.

Note that the ending level is uniquely defined if a heavy cell identification scheme is

fixed. We now rewrite the telescope sum for p as follows,

$$p = \sum_{i=0}^{l(p)} \left(c_p^i - c_p^{i-1}\right) + c_p^L - c_p^{l(p)},$$

where $c_p^{-1} = \mathbf{0}$ and $c_p^L = p$. For arbitrary k-centers $Z \subset [\Delta]^d$, we write,

$$d(p, Z) = \textstyle\sum_{i=0}^{l(p)} \left(d(c_p^i, Z) - d(c_p^{i-1}, Z)\right) + d(c_p^L, Z) - d(c_p^{l(p)}, Z) + d(\mathbf{0}, Z)$$

Let P_l be all the points with ending level $l(p) = l$. We now present the following lemmas.

Lemma 4.7.3. Let P_i be the set of points with ending level i. Let $Z^* \subset [\Delta]^d$ be a set of optimal k-centers for the k-median problem of the input point set. Assume that for each $i \in [-1, L]$, at most $ek(L+1)/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \leq \Delta/(2^{i+1}d)$. Then

$$|P_i| \cdot \frac{\Delta\sqrt{d}}{2^i} \leq \lambda_2 d^{3/2} \mathsf{OPT},$$

where $\lambda_2 > 0$ is a universal constant.

Before we prove this lemma, we first introduce the following lemmas to bound the cells with a large number of points.

Lemma 4.7.4. Assume that for each $i \in [0, L]$, at most $ek(L+1)/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \leq \Delta/(2^{i+1}d)$. Then for any $r > 0$ there are at most $\frac{(e+2r)k(L+1)}{\rho}$ cells that satisfy $|\mathcal{C}| \geq \frac{2^i \rho d \mathsf{OPT}}{rk(L+1)\Delta}$.

Proof of Lemma 4.7.4. Let $L' = L + 1$. Fix a value $i \in [0, L]$ and then $W = \Delta/2^i$ is the width of a cell in $\mathcal{G}_i$. Since at most ekL/ρ cells in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \leq W/(2d)$, then of the

remaining cells, each contribute at least $\frac{\rho d\mathsf{OPT}}{rWkL'}\frac{W}{2d} = \frac{\rho\mathsf{OPT}}{2rkL'}$ to the cost, and the cost of these cells is at most OPT. Therefore there can be at most $2rL'k/\rho$ cells such that $d(\mathcal{C}, Z^*) > W/(2d)$. Along with the at most ekL'/ρ cells (by the assumption) such that $d(\mathcal{C}, Z^*) \leq W/(2d)$, there are at most $(e+2r)L'k/\rho$ cells that contain at least $\rho d\mathsf{OPT}/(rWkL')$ points.

□

Lemma 4.7.5. Assume that for each $i \in [0, L]$, at most $ek(L+1)/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \leq \Delta/(2^{i+1}d)$. Then for $i \in [-1, L]$, the points of P_i can be partitioned to at most $k' = \frac{2(e+6)k(L+1)}{\rho}$ groups, $G_1, G_2, \ldots, G_{k'}$, such that for each $j \in [k']$, there exists a $\mathcal{C} \in \mathcal{G}_i$, such that $G_j \in \mathcal{C}$, $|G_j| < 5\frac{2^{i-1}\rho d\mathsf{OPT}}{k(L+1)\Delta}$.

Proof of Lemma 4.7.5. Let $L' = L + 1$. For each heavy cell in $\mathcal{G}_i$, if the number of points falling into its non-heavy subcells (in $\mathcal{G}_{i+1}$) is less than $\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}$, we group all these subcells into a single group. Let the groups formed this way be called type I, and by Property 4 of Definition 4.4.1 there are at most $(e+4)kL'/\rho$ type I groups.

For each of the remaining heavy cells in $\mathcal{G}_i$, we group its subcells into groups such that each group contains a number of points in the interval $\left[\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}, 5\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}\right)$. This can be done since each non-heavy subcell contains less than $\frac{2^{i+1}\rho d\mathsf{OPT}}{kL'\Delta} = 4\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}$ points, and the total number of points contained in them is at least $\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}$ (otherwise we would have formed a type I group). Let the groups formed this way be called type II. By the assumption of of Lemma 4.7.3, at most $\frac{ekL'}{\rho}$ of these non-heavy subcells are within distance $\frac{\Delta}{2^{i+2}d}$ from an optimal center of Z^*. Since each type II group contains at least $\frac{2^{i-1}\rho d\mathsf{OPT}}{kL'\Delta}$ points, by the same argument as in Lemma 4.7.4, the number of type II groups further than

distance $\frac{\Delta}{2^{i+2}d}$ from an optimal center is at most $\frac{8kL'}{\rho}$. We conclude that,

$$k' \leq \frac{(e+4)kL'}{\rho} + \frac{(e+8)kL'}{\rho}. \qquad \square$$

Proof of Lemma 4.7.3. Let $L' = L + 1$. Fix a value $i \in [-1, L]$ and then $W = \Delta/2^i$ is the upper bound of the width of a cell in $\mathcal{G}_i$. Let $G_1, G_2, \ldots G_{k'}$ be group of points satisfying Lemma 4.7.5. Thus, $\sum_{p \in P_l} \frac{\Delta\sqrt{d}}{2^i} \leq \sum_{j \in [k']} \frac{2^{i+1}\rho d\mathsf{OPT}}{kL'\Delta} \cdot \frac{\Delta\sqrt{d}}{2^i} \leq \frac{\lambda' kL'}{\rho} \cdot \frac{2^{i+1}\rho d\mathsf{OPT}}{kL'\Delta}\frac{\Delta\sqrt{d}}{2^i} \leq \lambda_2 d^{3/2}\mathsf{OPT}$ for some universal constants λ' and λ_2. $\square$

Proof of Proposition 4.7.10. First notice that the weighted set satisfies the about condition is an ϵ-coreset. If we replace each $|\widehat{\mathcal{C}_i}|$ by the exact number of points in $|\mathcal{C}_i|$, then the new weighted set is an $(\epsilon/2)$-coreset. For each $\mathcal{C} \in \mathcal{G}$, let $b_{\mathcal{C}}$ be the new value returned by the algorithm, and b_q is the new value of a point $q \in S$. The error of the cost introduced is at most,

$$A = \sum_{i=0}^{L} \left(\sum_{\mathcal{C} \in \mathcal{G}_i:\ \text{heavy}} |\widehat{|\mathcal{C}|} - b_{\mathcal{C}}| + \sum_{p \in S_{i-1}} \left| \left(\frac{1}{\pi_{i-1}} - b_p \right) \right| \right) \frac{\Delta\sqrt{d}}{2^i}.$$

By the procedure, the new value of a cell is always smaller than its original value, thus

$$A = \sum_{i=0}^{L} \left(\sum_{\mathcal{C} \in \mathcal{G}_i:\ \text{heavy}} \widehat{|\mathcal{C}|} - b_{\mathcal{C}} + \sum_{p \in S_{i-1}} \left(\frac{1}{\pi_{i-1}} - b_p \right) \right) \frac{\Delta\sqrt{d}}{2^i} = \sum_{i=0}^{L} g_i,$$

where

$$g_i = \left(\sum_{\mathcal{C} \in \mathcal{G}_i:\text{heavy}} \widehat{|\mathcal{C}|} - b_{\mathcal{C}} + \sum_{p \in S_{i-1}} \frac{1}{\pi_{i-1}} - b_p \right) \frac{\Delta\sqrt{d}}{2^i}.$$

Let

$$f_i = \left(\sum_{\mathcal{C} \in \mathcal{G}_i : \text{heavy}} \left| |\mathcal{C}| - \widehat{|\mathcal{C}|} \right| + \sum_{\mathcal{C}' \in \mathcal{G}_{i-1} : \text{heavy}} \left| \frac{|S_{i-1} \cap \mathcal{C}'|}{\pi_{i-1}} - |P_{i-1} \cap \mathcal{C}'| \right| \right) \frac{\Delta\sqrt{d}}{2^i}.$$

Thus $f_i \leq \epsilon \mathsf{OPT}/L$ by choosing appropriate λ_6. Now consider heavy cell $\mathcal{C} \in \mathcal{G}_i$, let $s_{\mathcal{C}} = \left| b_{\mathcal{C}} - \sum_{\mathcal{C} \in \mathcal{G}_{i+1} : \text{heavy}} \widehat{|\mathcal{C}|} - \frac{|S_i \cap \mathcal{C}|}{\pi_i} \right|$. Then,

$$\begin{aligned} s_{\mathcal{C}} &= \left| b_{\mathcal{C}} - \widehat{|\mathcal{C}|} + \widehat{|\mathcal{C}|} - |\mathcal{C}| - \sum_{\mathcal{C}' \in \mathcal{G}_{i+1} : \text{heavy}} (\widehat{|\mathcal{C}'|} - |\mathcal{C}'|) - \left(\frac{|S_i \cap \mathcal{C}|}{\pi_i} - |P_i \cap \mathcal{C}| \right) \right| \\ &\leq \left| b_{\mathcal{C}} - \widehat{|\mathcal{C}|} \right| + \left| \widehat{|\mathcal{C}|} - |\mathcal{C}| \right| + \sum_{\mathcal{C}' \in \mathcal{G}_{i+1} : \text{heavy}} \left| \widehat{|\mathcal{C}'|} - |\mathcal{C}'| \right| + \left| \frac{|S_i \cap \mathcal{C}|}{\pi_i} - |P_i \cap \mathcal{C}| \right|. \end{aligned} \tag{4.7.1}$$

Then

$$g_i = \sum_{\mathcal{C} \in \mathcal{G}_{i-1}} s_{\mathcal{C}} \frac{\Delta\sqrt{d}}{2^i} + \left(\sum_{p \in S_{i-1}} \frac{1}{\pi_{i-1}} - b_p \right) \frac{\Delta\sqrt{d}}{2^i} \leq \frac{1}{2} g_{i-1} + \frac{1}{2} f_{i-1} + f_i. \tag{4.7.2}$$

Since $g_{-1} = f_{-1} = 0$, thus

$$g_i \leq f_i + 3 \sum_{j=0}^{i-1} 2^{j-i} f_j, \quad \text{and} \quad \sum_{i=0}^{L} g_i \leq \sum_{i=0}^{L} f_i (1 + \sum_{j=1}^{i} \frac{3}{2^j}) \leq 4 \sum_{i=1}^{L} f_i \leq 4\epsilon \mathsf{OPT}. \qquad \square$$

Remark 4.7.6. The multiset of centers of heavy cells with each assigned a weight of the number of points in the cell is a $O(d^{3/2})$-coreset. This can be easily seen by removing the term of $d(c_p^L, Z) - d(c_p^{l(p)}, Z)$ from Equation (4.7.1) together with Lemma 4.7.3, which bounds the error introduced by this operation.

4.7.2 The New Construction (with arbitrary weights)

For these heavy cells, we use HEAVY-HITTER algorithms to obtain accurate estimates of the number of points in these cells, thus providing a *heavy cell identification scheme*. For the non-heavy cells, we only need to sample points from the bottom level, $\mathcal{G}_L$, but with a different probability for points with different ending levels. We present the following lemma that governs the correctness of sampling from the last level.

Lemma 4.7.7. If a set of points $P_i \subset P$ satisfies $|P_i|\Delta\sqrt{d}/(2^i) \leq \beta\mathsf{OPT}$ for some $\beta \geq 2\epsilon/(3(L+1))$, let S_i be an independent sample from P_i such that $p \in S_i$ with probability

$$\pi_i \geq \min\left(\frac{3a(L+1)^2\Delta\sqrt{d}\beta}{2^i\epsilon^2 o}\ln\frac{2\Delta^{kd}(L+1)}{\rho}, 1\right)$$

where $0 < o \leq a\mathsf{OPT}$ for some $a > 0$. Then for a fixed set $Z \subset [\Delta]^d$, with probability at least $1-\rho/((L+1)Delta^{kd})$, $|\sum_{p\in S_i}(d(c_p^i, Z) - d(p, Z))/\pi_i - \sum_{p\in P_i}(d(c_p^i, Z) - d(p, Z))| \leq \frac{\epsilon\mathsf{OPT}}{L+1}$.

Proof. The proof is identical to that of Lemma 4.3.4. □

Lemma 4.7.8. Consider a set of sets $\{P_i\}_{i=0}^L$ which satisfies $|P_i|\Delta\sqrt{d}/(2^i) \leq \frac{\beta\rho}{k(L+1)}\mathsf{OPT}$ for some $\beta \geq \epsilon/(3(L+1))$. For each $i \in [0, L]$, let S_i be an independent sample from P_i with sampling probability

$$\pi_i \geq \min\left(\frac{4a\beta k(L+1)^3\Delta\sqrt{d}}{2^i\epsilon^2\rho o}\log\frac{2}{\delta}, 1\right)$$

where $0 < o \leq a\mathsf{OPT}$ for some $a > 0$, then with probability at least $1 - \delta$,

$$\left| \frac{|S_i \cap P_i|}{\pi_i} - |P_i| \right| \frac{\Delta\sqrt{d}}{2^i} \leq \frac{\epsilon\rho\mathsf{OPT}}{k(L+1)^2}.$$

Proof of Lemma 4.7.8. The proof is simply by Bernstein inequality. Let $t = \frac{2^i\epsilon\rho\mathsf{OPT}}{\sqrt{d}k(L+1)^2\Delta}$, $X_p := \mathbb{I}_{p\in S_i}/\pi_i$, then we have that $Var(X_p) \leq 1/\pi_i$ and $b := \max_p |X_p| \leq 1/\pi_i$. By Bernstein's inequality, for any $j \in [k']$,

$$Pr\left[\left| \frac{|P_j \cap S_i|}{\pi_i} - |P_j| \right| > t\right] \leq 2e^{-\frac{t^2}{2|\mathcal{C}|/\pi_i + 2bt/3}} \leq \delta. \qquad \square$$

We now describe the new construction. This essentially has the same gaurantee as the simpler construction from the previous section, however the benefit here is that (as shown in the next subsection) it can be modified to output only positive weights. In the following paragraph, the estimations $\widehat{|\mathcal{C}|}$ are given as a blackbox. In proposition 4.7.9 we specify the conditions these estimations must satisfy.

Non-Negatively Weighted Construction Fix an arbitrary heavy cell identification scheme $\mathcal{H}$. Let P_l be all the points with ending level $l(p) = l$. For each heavy cell $\mathcal{C}$, let $\widehat{|\mathcal{C}|}$ be an estimation of number of points of $|\mathcal{C}|$, we also call $\widehat{|\mathcal{C}|}$ the *value* of cell $\mathcal{C}$. For each non-heavy cell $\mathcal{C}'$, let $\widehat{|\mathcal{C}'|} = 0$. Let S be a set samples of P constructed as follows: $S = S_{-1} \cup S_0 \cup S_1, \cup \ldots \cup S_L$, where S_l is a set of i.i.d samples from P_l with probability π_l. Here π_l for $l \in [-1, L]$ is redefined as,$\pi_l = \min\left(\frac{\lambda_3 d^2\Delta L^2}{2^l\epsilon^2 o}\log\left(\frac{2L\Delta^{dk}}{\rho}\right) + \frac{\lambda_4 d^2 k L^3\Delta}{2^i\epsilon^2\rho o}\log\frac{30kL^2}{\rho^2}, 1\right)$ where $\lambda_3 > 0$ and $\lambda_4 > 0$ are universal constants. Our coreset $\mathcal{S}$ is composed by all the

sampled points in S and the cell centers of heavy cells, with each point p assigned a weight $1/\pi_{l(p)}$ and for each cell center c of a heavy cell $\mathcal{C} \in \mathcal{G}_i$, the weight is,

$$\mathrm{wt}(c) = \widehat{|\mathcal{C}|} - \sum_{\substack{\mathcal{C}' : \mathcal{C}' \in \mathcal{G}_{i+1}, \mathcal{C}' \subset \mathcal{C}, \\ \mathcal{C}' \text{ is heavy}}} \widehat{|\mathcal{C}'|} - \frac{|S_i \cap \mathcal{C}|}{\pi_i}. \tag{4.7.3}$$

For each non-heavy cell $\mathcal{C}$ except for those in the bottom level, $\mathrm{wt}(c(\mathcal{C})) = 0$. The weight of each point from S is the value of the corresponding cell in the bottom level.

We now state the following proposition for a coreset construction, which immediately serves as an offline coreset construction.

Proposition 4.7.9. Let $\mathcal{H}$ be an arbitrary heavy cell identification scheme. Fix $\Omega(\Delta^{-d}) \leq \rho < 1$ and for each heavy $\mathcal{C} \in \mathcal{G}_i$ in level i, $\widehat{|\mathcal{C}|}$ is an estimation of number of points in $\mathcal{C}$ with additive error at most $\frac{\epsilon}{\lambda_5 L d^{3/2}} \cdot \frac{2^i \rho d \mathsf{OPT}}{kL\Delta}$, where $\lambda_5 > 0$ is a universal constant. Let S_l be the set of i.i.d. samples of P_l with probability $\pi_l(o)$. If $0 < o \leq \mathsf{OPT}$, then with probability at least $1 - 4\rho$, for every k-set $Z \subset [\Delta]^d$,

$$\left| \sum_{q \in \mathcal{S}} \mathrm{wt}(q) d(q, Z) - \sum_{p \in P} d(p, Z) \right| \leq \epsilon \mathsf{OPT}.$$

And the coreset size $|\mathcal{S}|$ is

$$O\left[\frac{d^3 L^4 k}{\epsilon^2} \left(d + \frac{1}{\rho} \log \frac{kL}{\rho} \right) \frac{\mathsf{OPT}}{o} \right].$$

Proof of Proposition 4.7.9. Fix a k-set $Z \subset [\Delta]^d$. First notice that,

$$\begin{aligned}
\widehat{\mathrm{cost}}(Z) &= \sum_{q \in \mathcal{S}} \mathrm{wt}(q) d(q, Z) \\
&= \sum_{i=-1}^{L-1} \left[\sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \text{ heavy}} \left(\widehat{|\mathcal{C}|} - \sum_{\substack{\mathcal{C}' : \mathcal{C}' \in \mathcal{G}_{i+1}, \mathcal{C}' \subset \mathcal{C}, \\ \mathcal{C}' \text{ is heavy}}} \widehat{|\mathcal{C}'|} - \frac{|S_i \cap \mathcal{C}|}{\pi_i} \right) d(c(\mathcal{C}), Z) + \sum_{p \in S_i} \frac{d(p, Z)}{\pi_i} \right] \\
&= \sum_{i=-1}^{L-1} \left[\sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \text{ heavy}} \widehat{|\mathcal{C}|} (d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P), Z)) + \sum_{p \in S_i} \frac{d(p, Z) - d(c_p^i, Z)}{\pi_i} \right],
\end{aligned} \tag{4.7.4}$$

where we denote $d(c(\mathcal{C}_{-1}^P), Z) = 0$ for convenience. Let $\mathrm{cost}(Z) = \sum_{p \in P} d(p, Z)$. Note that we can also write the true cost of Z as

$$\mathrm{cost}(Z) = \sum_{i=-1}^{L-1} \left[\sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \text{ heavy}} |\mathcal{C}| (d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P, Z))) + \sum_{p \in P_i} d(p, Z) - d(c_p^i, Z) \right].$$

We have that,

$$\widehat{\mathrm{cost}}(Z) - \mathrm{cost}(Z) = A_1 + A_2,$$

where

$$A_1 = \sum_{i=-1}^{L-1} \left[\sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \text{ heavy}} (\widehat{|\mathcal{C}|} - |\mathcal{C}|)(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P, Z))) \right]$$

and

$$A_2 = \sum_{i=-1}^{L-1} \left(\sum_{p \in S_i} \frac{d(p, Z) - d(c_p^i, Z)}{\pi_i} - \sum_{p \in P_i} d(p, Z) - d(c_p^i, Z) \right).$$

Let $Z^* \subset [\Delta]^d$ be a set of optimal k-centers for the k-median problem of the input point set. By Lemma 4.2.2, with probability at most $1 - \rho$, for each $i \in [0, L]$, if at most

$ek(L+1)/\rho$ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(\mathcal{C}, Z^*) \leq \Delta/(2^{i+1}d)$. Conditioning on this event, we have that, by Lemma 4.7.4 there are at most $k' = O\left(\frac{kL}{\rho}\right)$ heavy cells per level. Since for each $\mathcal{C} \in \mathcal{G}_i$, $\left|\widehat{|\mathcal{C}|} - |\mathcal{C}|\right| \leq \frac{\epsilon}{\lambda_5 L d^{3/2}} \cdot \frac{2^i \rho d \mathsf{OPT}}{kL\Delta}$, by choosing appropriate constant $\lambda_5 > 0$ we have

$$\begin{aligned} |A_1| &\leq \sum_{i=-1}^{L-1} \left| \sum_{\mathcal{C} \in \mathcal{G}_i : \mathcal{C} \text{ heavy}} (\widehat{|\mathcal{C}|} - |\mathcal{C}|)(d(c(\mathcal{C}), Z) - d(c(\mathcal{C}^P, Z))) \right| \\ &\leq L \cdot k' \cdot \frac{\epsilon}{\lambda_5 L d^{3/2}} \cdot \frac{2^i \rho d \mathsf{OPT}}{kL\Delta} \cdot \frac{\Delta\sqrt{d}}{2^i} \leq \frac{\epsilon \mathsf{OPT}}{2}. \end{aligned} \tag{4.7.5}$$

For A_2, let

$$A_{2i} = \left(\sum_{p \in S_i} \frac{d(p, Z) - d(c_p^i, Z)}{\pi_i} - \sum_{p \in P_i} d(p, Z) - d(c_p^i, Z) \right).$$

By Lemma 4.7.3, for each $i \in [-1, L-1]$, $|P_i|\Delta\sqrt{d}/(2^i) = \lambda_2(d^{3/2}\mathsf{OPT})$. Thus by Lemma 4.7.7, and choosing appropriate constants, with probability at least $1 - \rho/(L+1)\Delta^{dk}$, $|A_{2i}| \leq \frac{\epsilon \mathsf{OPT}}{2(L+1)}$. By the union bound, with probability at least $1-\rho$, for every level i, and every k-set $Z \subset [\Delta]^d$, $|A_{2i}| \leq \frac{\epsilon \mathsf{OPT}}{2(L+1)}$. Thus $|A_2| \leq \epsilon \mathsf{OPT}/2$. In total, with probability at least $1 - 3\rho$, $|A_1 + A_2| \leq \epsilon \mathsf{OPT}$ for any k-set $Z \subset [\Delta]^d$.

The coreset size is the number of heavy cells plus the number of sampled points. The number of heavy cells is $O(kL^2/\rho)$. The expected number of sampled points per level is at most,

$$|S_i| = O\left(\frac{d^4 L^3 k}{\epsilon^2} + \frac{d^3 L^2 k}{\epsilon^2 \rho} \log\left(\frac{kL}{\rho}\right) \right) \frac{\mathsf{OPT}}{o}.$$

By a Chernoff bound, with probability at least $1 - \rho/\Delta^{dk}$, for every level $i \in [0, L]$, the

number of sampled points is,

$$|S_i| \leq O\left(\frac{d^4L^3k}{\epsilon^2} + \frac{d^3L^3k}{\rho\epsilon^2}\log\left(\frac{kL}{\rho}\right)\right)\frac{\mathsf{OPT}}{o}.$$

Thus the size of the coreset $\mathcal{S}$ is,

$$|\mathcal{S}| \leq O\left[\frac{d^3L^4k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\frac{\mathsf{OPT}}{o}\right].$$

□

4.7.3 Ensuring Non-Negative Weights

In this section, we will provide a procedure to rectify all the weights for the coreset constructed in the last sub-section. The idea is similar to the method used in.[115] The procedure is shown in Algorithm 8.

Proposition 4.7.10. Let $\mathcal{S}$ be a weighted set constructed using the Non-Negatively Weighted Construction, i.e. each heavy cell $\mathcal{C}$ has value $\widehat{|\mathcal{C}|}$ and the set of sampled points $S = S_{-1} \cup S_0 \ldots \cup S_L$ with each point in S_l has weight $1/\pi_l$. If for each heavy cell $\mathcal{C} \in \mathcal{G}_i$, $||\widehat{\mathcal{C}}| - |\mathcal{C}|| \leq \frac{\epsilon}{\lambda_6 L d^{3/2}} \cdot \frac{2^i \rho d \mathsf{OPT}}{kL\Delta}$ for some universal constant $\lambda_6 > 0$ and for each $i \in [-1, L]$ and any k-set $Z \subset [\Delta]^d$,

$$\left|\sum_{p\in S} \mathrm{wt}(p)(d(c_p^i, Z) - d(p, Z)) - \sum_{p\in P_i}(d(c_p^i, Z) - d(p, Z))\right| \leq \frac{\epsilon\mathsf{OPT}}{2L},$$

and

$$\sum_{\mathcal{C}\in\mathcal{G}_i:\text{ heavy}} \left| \frac{|S_i \cap \mathcal{C}|}{\pi_i} - |P_i \cap \mathcal{C}| \right| \frac{\Delta\sqrt{d}}{2^i} \leq \frac{\epsilon \mathsf{OPT}}{L}.$$

Then on input $\widehat{|\mathcal{C}_1|}, \widehat{|\mathcal{C}_2|}, \ldots, \widehat{|\mathcal{C}_{k'}|}$ and S, where k' is the number of heavy cells, the coreset output by Algorithm 8 is a (4ϵ)-coreset.

4.7.4 The Streaming Algorithm

Sampling From Sparse Cells

For the streaming algorithm, we can still use `HEAVY-HITTER` algorithms to find the heavy cells. The major challenge is to do the sampling for each point from its ending level. We do this using a combination of hash functions and `K-Set`. In Algorithm 9, we provide a procedure that recovers the set of points from cells with a small number of points and ignore all the heavy cells. The guarantee is,

Theorem 4.7.11. Given as input a set of dynamically updating streaming points $P \subset [N]$, a set of mutually disjoint cells $C \subset [M]$, whose union covers the region of P. Algorithm 9 outputs all the points in cells with less than β points or output `Fail`. If with the promise that at most α cells from C contain a point of P, then the algorithm outputs `Fail` with probability at most δ. The algorithm uses $O(\alpha\beta(\log(M\beta) + \log N)\log N \log(\log N\alpha\beta/\delta)\log(\alpha\beta/\delta))$ bits in the worst case.

The high level idea of this algorithm is to hash the original set of points to a universe of smaller size. For cells with less points, the collision rate is much smaller than cells with more

points. To recover one bit of a point, we update that bit and the cell ID and also its hash tag to the `K-Set`-data structure. If there are no other points with hash values colliding with this point, the count of that point is simply 1. If this is the case, we immediately recover the bit. By repeating the above procedure once for each bit, we can successfully recover the set of points with no colliding hash tags. For those points with colliding hash tags, we simply ignore them. Each point has a constant probability to collide with another point, thus not be in the output. By running the whole procedure $O(\log(\alpha\beta/\epsilon))$ times in parallel, we reduce the probability to roughly ϵ for each point in cells with less than β points. To formally prove Theorem 4.7.11, we first prove the following lemma, which is the guarantee of Algorithm 10.

Lemma 4.7.12. Given input a set of dynamically updating streaming points $P \subset [N]$, a set of mutually disjoint cells $C \subset [M]$, whose union covers the region of P. Algorithm 9 outputs a set of points in cells with less than β points or output `Fail`. If with the promise that at most α cells from C contain a point of P, then the algorithm outputs `Fail` with probability at most δ. Conditioning on the event that the algorithm does not output `Fail`, each point p from cell with less than β points is in the output with marginal probability at least 0.9. The algorithm uses $O(\alpha\beta(\log(M\beta) + \log N)\log N \log(\log N\alpha\beta/\delta))$ bits in the worst case.

Proof. We prove this lemma by showing that (a) if a point $p \in P$ contained in cell $\mathcal{C}$, with $|\mathcal{C} \cap P| \leq \beta$, then with probability at least 0.99, there are no other points $p' \in \mathcal{C} \cap P$ with $H(p) = H(p')$, (b) conditioning on the event that the algorithm does not output `Fail`, then for any cell $\mathcal{C} \in C$, if a point $p \in \mathcal{C}$ such that no other points in $\mathcal{C} \cap P$ has the same hash

value $H(p)$, then p is in the output and (c) the algorithm outputs `Fail` with probability at most δ. The correctness of the algorithm follows by (a), (b) and (c).

To show (a), consider any cell $\mathcal{C} \in C$ with $|\mathcal{C}| \leq \beta$, let $p \in \mathcal{C}$ with hash value $H(p)$. Since H is 2-wise independent, the expected number of other points hashed to the same hash value $H(p)$ is at most $\beta/U = 1/100$. By Markov's inequality, with probability at least 0.99, no other point in $\mathcal{C}$ is hashed to $H(p)$.

To show (b), notice that if the algorithm does not output `Fail`, then for a given cell $\mathcal{C}$, let c be its ID, and p_j be the j-th bit of point p. Then (c, h, p_j) has 1 count and $(c, h, 1-p_j)$ has 0 count for each $j \in [t]$, where $t = \lceil \log N \rceil$. Thus we can uniquely recover each bit of point p, hence the point p.

For (c), since there are at most α cells, there are at most $2\alpha U = O(\alpha\beta)$ many different updates for each `KS` structure. Therefore, with probability at most $\frac{\delta}{t}$, a single `KS` instance outputs `Fail`. By the union bound, with probability at least $1 - \delta$, no `KS` instance outputs `Fail`.

Finally, the space usage is dominated by the `KS` data structures. Since the input data to `KS` is from universe $[M] \times [U] \times \{0, 1\}$, each `KS` structure uses space $O(\alpha\beta(\log(M\beta) + \log N)\log(t\alpha\beta/\delta))$ bits of memory, the total space is $O(\alpha\beta(\log(M\beta)+\log N)\log N \log(t\alpha\beta/\delta))$.

□

Proof of Theorem 4.7.11. Each instance of `SparseCellsSingle` fails with probability at most $\delta/(4A)$, where A is the number independent `SparseCellsSingle` instances. By the

union bound, with probability at least $1-\delta/4$, none of them output `Fail`. Conditioning on this event, the random bits of the hash functions of each `SparseCellsSingle` instance are independent, thus by Lemma 4.7.12 a fixed point $p \in \mathcal{C}$ with $|\mathcal{C}| \leq \beta$ is in the output with probability at least $10^{-\log \frac{4\alpha\beta}{\delta}} \leq \delta/(4\alpha\beta)$. Since there are at most $\alpha\beta$ points in cells with less than β points, by the union bound we conclude that with probability at least $1-\delta/4$, every point in cells with less than β points is in the output set S. In sum, with probability at least $1-\delta/2$, S contains all the desired points.

The other `KS` instance outputs `Fail` with probability at most $\delta/2$. Thus if T is not `Fail`, then T contains the exact number of points of each cell. If any desired point is not in S, then $|\mathcal{C}| > |\mathcal{C} \cap S|$, we output `Fail`. This happens with probability at most δ under the gaurantee of the `KS`structure.

Since each `SparseCellsSingle` instance uses $O(\alpha\beta(\log(M\beta)+\log N)\log N \log(tA\alpha\beta/\delta))$ bits of space, the final space of the algorithm is

$$O(\alpha\beta(\log(M\beta)+\log N)\log N \log(t\alpha\beta/\delta)\log(\alpha\beta/\delta))$$

. □

The Algorithm

With the construction of algorithm `SparseCells`, we now have all the tools for the streaming coreset construction. The streaming algorithm is composed by $O(L)$ levels

of `HEAVY-HITTER` instances, which serve as a heavy cell identifier and by $O(L)$ levels of `SparseCells` instances, which sample the points from their ending levels. The full algorithm is stated in Algorithm 11. The guarantee of the algorithm is stated in the following theorem.

Theorem 4.7.13. Fix $\epsilon, \rho \in (0, 1/2)$, positive integers k and Δ, Algorithm 11 makes a single pass over the streaming point set $P \subset [\Delta]^d$, outputs a weighted set S with *non-negative weights* for each point, such that with probability at least 0.99, S is an ϵ-coreset for k-median of size $O\left[\frac{d^3L^4k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\right]$, where $L = \log\Delta$. The algorithm uses

$$O\left[\frac{d^7L^7k}{\epsilon^2}\left(\rho dL + \frac{1}{\rho}\log^2\frac{dkL}{\rho\epsilon}(\log\log\frac{dkL}{\rho\epsilon} + L)\right)\log^2\frac{dkL}{\rho\epsilon}\right]$$

bits in the worst case. For each update of the input, the algorithm needs poly $(d, 1/\epsilon, L, \log k)$ time to process and outputs the coreset in time poly$(d, k, L, 1/\epsilon, 1/\rho, \log k)$ after one pass of the stream.

Proof. W.l.o.g. assume $\rho \geq \Delta^{-d}$, since otherwise we can store the entire set of points. In the sequel, we will prove the theorem with parameter $O(\rho)$ and $O(\epsilon)$. It translates to ρ and ϵ directly by scaling and with losing at most a constant factor in space and time bounds. By Lemma 4.2.2, with probability at least $1 - \rho$, for every level $i \in [0, L]$, at most ekL/ρ cells $\mathcal{C}$ in $\mathcal{G}_i$ satisfy $d(C, Z^*) \leq \Delta/(2^{i+1}d)$. We condition on this event for the following proof.

We first show that the `HEAVY-HITTER` instances faithfully implement a *heavy cell identification scheme*. First note that with probability at least $1 - \rho$, all `HEAVY-HITTER` instances

succeed. Conditioning on this event for the following proof. As shown in the proof of Lemma 4.6.1, by setting $\epsilon' = \epsilon\sqrt{\frac{\rho}{\lambda_7 k d^3 L^3}}$ and $k' = \lambda_8 kL/\rho$, for appropriate positive universal constants λ_7, λ_8, then the additive error to each cell is at most $\frac{\epsilon}{\lambda_9 d^{3/2} L} \cdot \frac{2^i d\mathsf{OPT}}{kL\Delta}$ for some universal constant λ_9, which matches the requirement of Proposition 4.7.10. For each cell $\mathcal{C}$ with at least $2^i \rho d\mathsf{OPT}/(k(L+1)\Delta)$ points, by Lemma 4.7.4 it must be in the top $(e+2)k(L+1)/\rho$ cells. For each cell $\mathcal{C}'$ with at least $2^{i-1}\rho d\mathsf{OPT}/(k(L+1)\Delta)$ points, it must be in the top $(e+4)k(L+1)/\rho$ cells. Since the additive error is $\frac{\epsilon}{\lambda_7 d^{3/2}(L+1)} \cdot \frac{2^i d\mathsf{OPT}}{k(L+1)\Delta} \ll \frac{1}{2}\frac{2^i d\mathsf{OPT}}{k(L+1)\Delta}$. Thus $\mathcal{C}$ is in the output of the `HEAVY-HITTER` instances, since otherwise $\widehat{|\mathcal{C}|} \leq \frac{1}{2}\frac{2^i d\mathsf{OPT}}{k(L+1)\Delta} + \frac{\epsilon}{\lambda_7 d^{3/2}(L+1)} \cdot \frac{2^i d\mathsf{OPT}}{k(L+1)\Delta}$ contradicts the error bound (by choosing sufficiently large λ_7). Thus the algorithm faithfully implements a heavy cell identification scheme.

Now we show that if there exists an $o \leq \mathsf{OPT}$ such that no instance of `SparseCells` outputs `Fail`, then the result is a desired $O(\epsilon)$-coreset. This follows by Proposition 4.7.9 and Proposition 4.7.10. Then we note that the coreset size is upper bounded by

$$O\left[\frac{d^3 L^4 k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\right]$$

as desired.

Next we show that there exists an $\mathsf{OPT}/2 \leq o^* \leq \mathsf{OPT}$ that with probability at least $1-\rho$, no `SparseCells` instance $\mathtt{SC}_{o^*,i}$ outputs `Fail`. By Chernoff bound, with probability at least $1 - O(\rho)$, as also shown in the proof of Proposition 4.7.9, per level at most

$O\left[\frac{d^3L^4k}{\epsilon^2}\left(d+\frac{1}{\rho}\log\frac{kL}{\rho}\right)\frac{\mathsf{OPT}}{o}\right]$ cells is occupied by a point. And at most

$$O\left[\frac{d^3L^2}{\epsilon^2}\left(\rho d+\log\frac{kL}{\rho}+\frac{\rho}{kL}\log\frac{L}{\rho}\right)\right]$$

points is sampled per light cell. Conditioned on this fact and that each `SC` instances fails with probability at most $O(\rho/(dL))$, with probability at least $1-O(\rho)$, no nstance $\mathtt{SC}_{o^*,i}$ fails.

Lastly, we bound the space usage and update/query time. For the `HEAVY-HITTER` instances, the total space used is $O\left(dL+\log\frac{1}{\rho}\right)\frac{d^4L^5k}{\rho\epsilon^2}$ bits, analogous to the proof of Theorem 4.3.6. Each `SparseCells` instance uses space $O\left[\frac{d^5L^4r^2k}{\epsilon^2}\left(\rho dL+\frac{r^2(\log r+L)}{\rho}\right)\right]$, where $r=\log\frac{dkL}{\rho\epsilon}$. The total space bound is $O\left[\frac{d^6L^6r^2k}{\epsilon^2}\left(\rho dL+\frac{r^2(\log r+L)}{\rho}\right)\right]$ bits. As a same argument in the proof of Theorem 4.3.6, the cost of de-randomization introduce an additional dL factor. Thus, the final space bound is $O\left[\frac{d^7L^7r^2k}{\epsilon^2}\left(\rho dL+\frac{r^2(\log r+L)}{\rho}\right)\right]$ bits. The query time and update time is similar to that of Theorem 4.3.6 thus poly $\left(d,L,\frac{1}{\epsilon},\frac{1}{\rho},k\right)$ and poly $\left(d,L,\frac{1}{\epsilon},\log k\right)$.

□

4.8 Experiments

We illustrate our construction using an offline construction on Gaussian mixture data in $\mathbb{R}^2$. As shown in Figure 4.1 in Section 4.8, we randomly generated 65536 points from $\mathbb{R}^2$, then rounded the points to a grid of size $\Delta=512$. Our coreset uses $\log_2\Delta+2=11$ levels

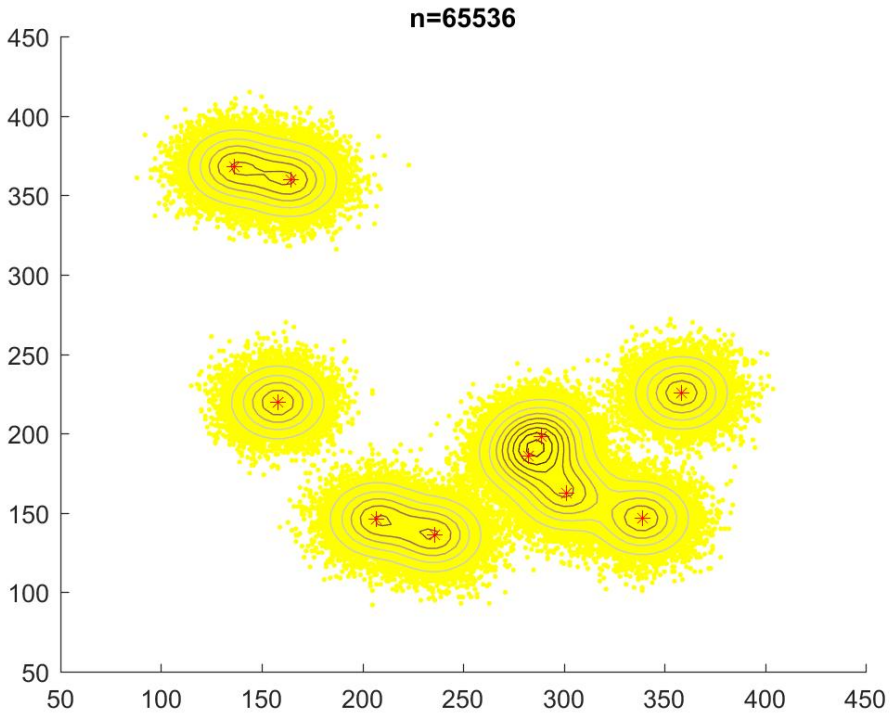
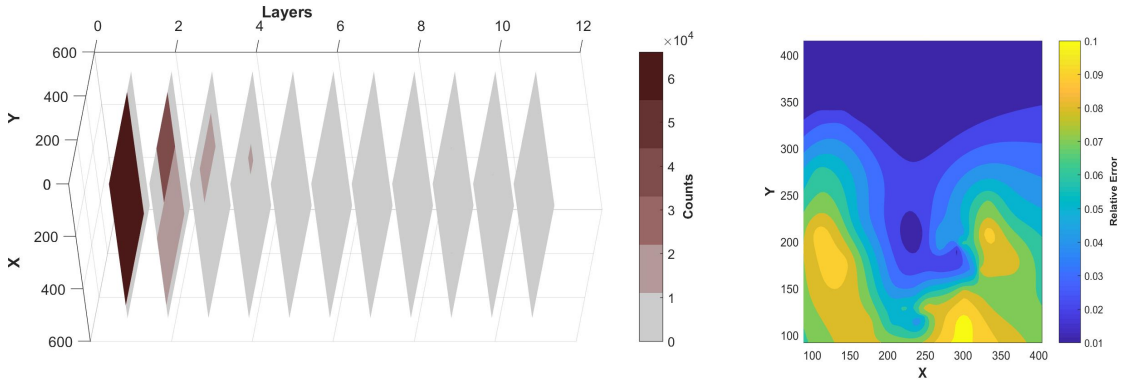


Figure 4.1: 65536 points are drawn from a Gaussian Mixture distribution. The contours illustrate the PDF function.

of grids. The storage in each level is very sparse. As shown in Figure 4.2(a), only 90 points are stored in total. We compared the 1-median costs estimated using the coreset and the dataset, the resulting difference is very small, as illustrated in Figure 4.2(b).

4.9 Concluding Remark

We develop algorithms that make a single pass over the dynamic stream and output, with high probability, a coreset for the original k-median problem. Both the space complexity and the size of the coreset are polynomially dependent on d, whereas the only previous known bounds are exponential in d. We constructed our coreset for the possible solutions in



(a) (b)

Figure 4.2: (a) The layer structure of the coreset. Cells with more weight are shaded darker. (b) The relative error of a 1-median cost function. Using only 90 points, the global maximum error was under 10%.

discrete space $[\Delta]^d$, but it is easy to modify the coreset to be a coreset in continuous space $[0, \Delta]^d$ (note that we still require the input dataset to be from a discrete space). The way to do this is by modifying the sampling probability π_i in the algorithm, i.e. replacing the factor of $\ln(\Omega(\Delta^{kd}L/\rho))$ to $\ln(\Omega((\Delta/\epsilon)^{kd}L/\rho))$. Then any k-set from $[0, \Delta]^d$ can be rounded to the closest k-set in $[\Delta/\epsilon]^d$ and the cost only differs by a $(1 \pm \epsilon)$ factor while the space bound changes only by a polylog$(1/\epsilon)$ factor. Lastly, we remark that the coreset scheme can be easily modified to other metric spaces, e.g. the l_p metric. The space bound depends on the doubling dimension of the metric.

As shown in our experiments, a 2D implementation using our framework is very efficient. We believe that a high-dimensional implementation will be efficient as well. We leave the full implementation as a future project.

Algorithm 6 `CoreSet`(S, k, ρ, ϵ): construct a ϵ-coreset for dynamic stream S.

Initization:
Initialize a grid structure;
$O \leftarrow \{1, 2, 4, \ldots, \sqrt{d}\Delta^{d+1}\}$;
$L \leftarrow \lceil \log \Delta \rceil$;
$\pi_i(o) \leftarrow \min\left(\frac{3(L+1)^2\Delta d^2}{2^i\epsilon^2 o} \ln \frac{2\Delta^{kd}(L+1)}{\rho}, 1\right)$;
$K \leftarrow \frac{(2+e)(L+1)k}{\rho} + \frac{24d^4(L+1)^3k}{\epsilon^2} \ln \frac{1}{\rho}$,
$\epsilon' \leftarrow \left(\epsilon\sqrt{\frac{\rho}{8(2+e)^2kd^3(L+1)^3}}\right)$; $m \leftarrow 0$;
For each $o \in O$ and $i \in [0, L]$, construct fully independent hash function $h_{o,i} : [\Delta]^d \to \{0, 1\}$ with $Pr_{h_{o,i}}(h_{o,i}[q] = 1) = \pi_i(o)$;
Initialize `K-Set` instances $\texttt{KS}_{o,i}$ with error probability $\rho/(L+1)$, size parameter K;
Initialize `HEAVY-HITTER`$(\Delta^d, (e+2)(L+1)k/\rho, \epsilon', \rho/(L+1))$ instances, $\texttt{HH}_0, \texttt{HH}_1, \ldots, \texttt{HH}_L$, one for a level;
Update (S):
for *each update* $(op, q) \in S$**:**
    /*$op \in \{$`Insert`, `Delete`$\}$*/
    $m \leftarrow m \pm 1$; /*`Insert`: $+1$, `Delete`:-1*/
    for *each* $i \in [0, L]$**:**
        $c_q^i \leftarrow$ the center of the cell contains q at level i;
        $\texttt{HH}_i$.update(op, c_q^i);
        for *each* $o \in [O]$**:**
            if $h_{o,i}(q) == 1$**:**
                $\texttt{KS}_{o,i}$.update(op, c_q^i);
Query:
Let o^* be the smallest o such that no instance of $\texttt{KS}_{o,0}, \texttt{KS}_{o,1}, \ldots, \texttt{KS}_{o,L}$ returns `Fail`;
$R \leftarrow \{\}$;
for $i = -1$ *to* L**:**
    for *each cell center* c *in level* i**:**
        Let $\mathcal{C}$ be the cell containing c;
        if i = *-1*:
            $f \leftarrow m$;
        else:
            $f \leftarrow$ `GetFreq`$(c, \texttt{HH}_i, \texttt{KS}_{o^*,i}, \pi_i(o^*))$;
        if $i < L$**:**
            $g \leftarrow \sum_{\mathcal{C}' \subset \mathcal{C}: \mathcal{C}' \in \mathcal{G}^{i+1}}$ `GetFreq` $(c(\mathcal{C}'), \texttt{HH}_{i+1}, \texttt{KS}_{o^*i+1}, \pi_i(o^*))$;
            Assign weight $f - g$ to c;
            if $f - g \neq 0$**:**
                $R \leftarrow R \cup \{c\}$;
        else:
            Assign weight f to c;
            if $f \neq 0$**:**
                $R \leftarrow R \cup \{c\}$;
return R.

Algorithm 7 `GetFreq`$(e, \texttt{HH}, \texttt{KS}, \pi_i)$: retrieve the correct freuquency of cell center e, given the instance of `HEAVY-HITTER` and K-set.

$f_S(e) \leftarrow$ the frequency of e returned by `HH`;
$f_K(e) \leftarrow$ the frequency of e returned by `KS`;
$k' \leftarrow (e+2)(L+1)k/\rho$;
$F \leftarrow$ the set of top-k' heavy hitters returned by `HEAVY-HITTER`;
if $e \in F$**:**
  return $f_S(e)$;
else:
  return $f_K(e)/\pi_i$.

Algorithm 8 `RectifyWeights`$\left(\widehat{|\mathcal{C}_1|}, \widehat{|\mathcal{C}_2|} \ldots, \widehat{|\mathcal{C}_{k'}|}, S\right)$: input the estimates of number of points in each cell and the weighted sampled points, output a weighted coreset with non-negative weights.

for $i = -1$ *to* L**:**
  for *each heavy cell* $\mathcal{C}$ *center in* $\mathcal{G}_i$**:**
    if $\mathrm{wt}(\mathcal{C}) < 0$**:**
      Decrease the value of the children heavy cells in level $\mathcal{G}_{i+1}$ and sampled points S_i arbitrarily by total $|\mathrm{wt}(\mathcal{C})|$ amount, such that for each children cell $\mathcal{C}' \in \mathcal{G}_{i+1}$, $\widehat{|\mathcal{C}'|}$ is non-negative, and for each sampled point $p \in S_i$, the weight is non-negative.
return *Rectified Coreset*

Algorithm 9 `SparseCells`$(N, M, \alpha, \beta, \delta)$: input the point sets $P \subset [N]$ and set of cells $C \subset [M]$ such that at most α cells containing a point, output the set of points in cells with less than β points.

Let $A \leftarrow \log \frac{4\alpha\beta}{\delta}$;
Let $R_1, R_2, \ldots, R_A$ be the results of independent instances of `SparseCellsSingle`$(N, M, \alpha, \beta, \delta/(4A))$ running in parallel;
Let T be the results of another parallel `KS` structure with space parameter α and error $\delta/2$ and with input as the cell IDs of points in P; /*T returns the exact counts of each cell*/
if *any of the data structures returns* `Fail`**:**
  return `Fail`;
Let $S \leftarrow R_1 \cup R_2 \cup \ldots R_A$;
if $\exists$ *set* $\mathcal{C} \in T$ *with* $|\mathcal{C}| \leq \beta$ *and* $|\mathcal{C}| \neq |\mathcal{C} \cap S|$**:**
  return `Fail`;
return S;

Algorithm 10 `SparseCellsSingle`$(N, M, \alpha, \beta, \delta)$: input the point sets $P \subset [N]$ and set of cells $C \subset [M]$ such that at most α cells containing a point, output the set of points in cells with less than β points.

Initization:
$U \leftarrow 100\beta$.
$t \leftarrow \lceil \log N \rceil$;
$H : [N] \to [U]$, 2-wise independent;
`K-Set` structures $\texttt{KS}_1, \texttt{KS}_2, \ldots \texttt{KS}_t$ with space parameter $2\alpha U$ and probability $\frac{\delta}{t}$;
Update(p, op)**:** /*$op \in \{\texttt{Insert}, \texttt{Delete}\}$*/
$c \leftarrow$ cell ID of p;
for $i \in [t]$**:**
    /*A point p is represented as $(p_1, p_2, \ldots, p_t)$*/;
    $p_i \leftarrow$ the i-th bit of point p;
    $\texttt{KS}_i$.update$((c, H(p), p_i), op)$;

Query:
if *for any* $i \in [t]$, $\texttt{KS}_i$ *returns* `Fail`**:**
    return `Fail`;

$S \leftarrow \emptyset$;
for *each* (c, h, p_1) *in the output of* $\texttt{KS}_1$**:**
    if $(c, h, p_j) \notin \texttt{KS}_j$ *for some* $j \in [t]$**:**
        /*A checking step, may not happen at all*/;
        return `Fail`;
    Let $s(c, h, p_j)$ be the counts of (c, h, p_j) in $\texttt{KS}_j$;
    if $s(c, h, p_j) = 1$ *and* $s(c, h, 1 - p_j) = 0$ *for each* $j \in [t]$**:**
        $p \leftarrow (p_1, p_2, \ldots, p_t)$;
        $S \leftarrow S \cup \{p\}$;
return S

Algorithm 11 `PositiveCoreSet`(S, k, ρ, ϵ): construct a ϵ-coreset for dynamic stream S.

Initization:

Initialize a grid structure;

$O \leftarrow \{1, 2, 4, \ldots, \sqrt{d}\Delta^{d+1}\}$; $L \leftarrow \lceil \log \Delta \rceil$;

$\pi_i(o) \leftarrow \min\left(\frac{\lambda_3 d^2 \Delta L^2}{2^i \epsilon^2 o} \log\left(\frac{2L\Delta^{dk}}{\rho}\right) + \frac{\lambda_4 d^2 k L^3 \Delta}{2^i \epsilon^2 \rho o} \log \frac{30kL^2}{\rho^2}, 1\right)$;

$\alpha \leftarrow O\left[\frac{d^3 L^4 k}{\epsilon^2}\left(d + \frac{1}{\rho}\log\frac{kL}{\rho}\right)\right]$, $\beta \leftarrow O\left[\frac{d^3 L^2}{\epsilon^2}\left(\rho d + \log\frac{kL}{\rho} + \frac{\rho}{kL}\log\frac{L}{\rho}\right)\right]$;

$\epsilon' \leftarrow \epsilon\sqrt{\frac{\rho}{\lambda_7 k d^3 L^3}}$; $m \leftarrow 0$;

For each $o \in O$ and $i \in [0, L]$, construct fully independent hash function $h_{o,i} : [\Delta]^d \to \{0, 1\}$ with $Pr_{h_{o,i}}(h_{o,i}[q] = 1) = \pi_i(o)$; initialize `SparseCells`$(\Delta^d, (1+2^i)^d, \alpha, \beta, O(\rho/(dL)))$ instances $\mathtt{SC}_{o,i}$;

Initialize `HEAVY-HITTER`$(\Delta^d, 10Lk/\rho, \epsilon', \rho/L)$ instances, $\mathtt{HH}_0, \mathtt{HH}_1, \ldots, \mathtt{HH}_{L-1}$, one for a level;

Update (S):

for *each update* $(op, q) \in S$**:**

/*$op \in \{$`Insert`, `Delete`$\}$*/

$m \leftarrow m \pm 1$; /*`Insert`: $+1$, `Delete`:-1*/

for *each* $i \in [0, L]$**:**

$c_q^i \leftarrow$ the center of the cell contains q at level i;

$\mathtt{HH}_i$.update(op, c_q^i);

for *each* $o \in [O]$**:**

if $h_{o,i}(q) == 1$**:**

$\mathtt{SC}_{o,i}$.update(op, c_q^i);

Query:

Let o^* be the smallest o such that no instance of $\mathtt{SC}_{o,0}, \mathtt{SC}_{o,1}, \ldots, \mathtt{SC}_{o,L}$ returns `Fail`;

$S \leftarrow \{\}$;

$\mathcal{C}_{-1} \leftarrow$ the cell of the entire space $[\Delta]^d$; $\widehat{|\mathcal{C}_{-1}|} \leftarrow m$;

for $i \in [0, L-1]$**:**

$C_i \leftarrow \mathtt{HH}_i$.query().top$((e+4)(L+1)k/\rho)$;

Remove cells $\mathcal{C}$ from C_i if $\mathcal{C}^P(\mathcal{C}) \notin C_{i-1}$, where $\mathcal{C}^P(\mathcal{C})$ is the parent cell of $\mathcal{C}$ in level $i-1$;

$B_i \leftarrow \mathtt{SC}_{o^*,i}$.query();

$S_i \leftarrow \{p \in B_i : \mathcal{C}(p, i-1) \in C_{i-1}$ AND $\mathcal{C}(p, i) \notin C_i\}$;

Each point in $\mathcal{S}_i$ receives weight $1/\pi_i(o^*)$;

$S \leftarrow S \cup S_i$;

$k' \leftarrow \sum_{i\in[0,L]} |C_i|$;

Let $\{\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_{k'}\} = \cup_{i\in[0,L]} C_i \cup \{\mathcal{C}_{-1}\}$ be the set of heavy cells;

Let $\left\{\widehat{|\mathcal{C}_1|}, \widehat{|\mathcal{C}_2|}, \ldots \widehat{|\mathcal{C}_{k'}|}\right\}$ be the estimated frequency of each cell;

$R \leftarrow$ `RectifyWeights`$(\widehat{|\mathcal{C}_1|}, \widehat{|\mathcal{C}_2|}, \ldots \widehat{|\mathcal{C}_{k'}|}, S)$;

return R.

Bibliography

[1] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang, "Data streaming algorithms for estimating entropy of network traffic," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1. ACM, 2006, pp. 145–156.

[2] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One sketch to rule them all: Rethinking network flow monitoring with univmon," in *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 2016, pp. 101–114.

[3] A. Andoni, R. Fagin, R. Kumar, M. Patrascu, and D. Sivakumar, "Corrigendum to efficient similarity search and classification via rank aggregation by ronald fagin, ravi kumar and d. sivakumar (proc. sigmod'03)," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 1375–1376.

[4] J. Nelson and D. P. Woodruff, "Fast manhattan sketches in data streams," in *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2010, pp. 99–110.

BIBLIOGRAPHY

[5] S. Tirthapura and D. Woodruff, "Rectangle-efficient aggregation in spatial data streams," in *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. ACM, 2012, pp. 283–294.

[6] J. Gama and M. M. Gaber, *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

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

[8] Z. Liu, N. Ivkin, L. Yang, M. Neyrinck, G. Lemson, A. Szalay, V. Braverman, T. Budavari, R. Burns, and X. Wang, "Streaming algorithms for halo finders," in *e-Science (e-Science), 2015 IEEE 11th International Conference on*. IEEE, 2015, pp. 342–351.

[9] W. Liu, B. Schmidt, G. Voss, and W. Muller-Wittig, "Streaming algorithms for biological sequence alignment on gpus," *IEEE transactions on parallel and distributed systems*, vol. 18, no. 9, 2007.

[10] R. Kienzler, R. Bruggmann, A. Ranganathan, and N. Tatbul, "Large-scale dna sequence analysis in the cloud: A stream-based approach." in *Euro-Par Workshops (2)*, 2011, pp. 467–476.

[11] B. Chandramouli, M. Ali, J. Goldstein, B. Sezgin, and B. S. Raman, "Data stream management systems for computational finance," *Computer*, vol. 43, no. 12, pp. 45–52, 2010.

[12] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review,"

ACM Sigmod Record, vol. 34, no. 2, pp. 18–26, 2005.

[13] Z. Allen-Zhu and Y. Li, "First efficient convergence for streaming k-pca: a global, gap-free, and near-optimal rate," *arXiv preprint arXiv:1607.07837*, 2016.

[14] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM, 1996, pp. 20–29.

[15] P. Indyk and D. Woodruff, "Optimal approximations of the frequency moments of data streams," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. ACM, 2005, pp. 202–208.

[16] P. Indyk, "Stable distributions, pseudorandom generators, embeddings, and data stream computation," *J. ACM*, vol. 53, no. 3, pp. 307–323, 2006.

[17] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar, "An information statistics approach to data stream and communication complexity," *J. Comput. Syst. Sci.*, vol. 68, no. 4, pp. 702–732, 2004.

[18] V. Braverman and R. Ostrovsky, "Zero-one frequency laws," in *Proceedings of the 42nd annual ACM Symposium on Theory of Computing*. ACM, 2010, pp. 281–290.

[19] V. Braverman and S. R. Chestnut, "Universal sketches for the frequency negative moments and other decreasing streaming sums," in *APPROX/RANDOM 2015*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 40. Schloss Dagstuhl, 2015, pp. 591–605.

BIBLIOGRAPHY

[20] V. Braverman, S. R. Chestnut, D. P. Woodruff, and L. F. Yang, "Streaming space complexity of nearly all functions of one variable on frequency vectors," in *Proceedings of the 35th ACM Symposium on Principles of Database Systems*. New York, NY, USA: ACM, 2016, pp. 261–276. [Online]. Available: http://doi.acm.org/10.1145/2902251.2902282

[21] J. Błasiok, V. Braverman, S. R. Chestnut, R. Krauthgamer, and L. F. Yang, "Streaming symmetric norms via measure concentration," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2017. New York, NY, USA: ACM, 2017, pp. 716–729. [Online]. Available: http://doi.acm.org/10.1145/3055399.3055424

[22] A. McGregor, "Graph mining on streams," in *Encyclopedia of Database Systems*. Springer, 2009, pp. 1271–1275.

[23] J. L. Bentley and J. B. Saxe, "Decomposable searching problems i. static-to-dynamic transformation," *Journal of Algorithms*, vol. 1, no. 4, pp. 301–358, 1980.

[24] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in *Foundations of computer science, 2000. proceedings. 41st annual symposium on*. IEEE, 2000, pp. 359–366.

[25] M. Charikar, L. O'Callaghan, and R. Panigrahy, "Better streaming algorithms for clustering problems," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, 2003, pp. 30–39.

[26] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan, "Maintaining variance and k-medians over data stream windows," in *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2003, pp. 234–243.

[27] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, "Approximating extent measures of points," *Journal of the ACM (JACM)*, vol. 51, no. 4, pp. 606–635, 2004.

[28] S. Har-Peled and S. Mazumdar, "On coresets for k-means and k-median clustering," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, 2004, pp. 291–300. [Online]. Available: http://doi.acm.org/10.1145/1007352.1007400

[29] S. Har-Peled and A. Kushal, "Smaller coresets for k-median and k-means clustering," in *Proceedings of the 21st ACM Symposium on Computational Geometry, Pisa, Italy, June 6-8, 2005*, 2005, pp. 126–134. [Online]. Available: http://doi.acm.org/10.1145/1064092.1064114

[30] K. Chen, "On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications," *SIAM Journal on Computing*, vol. 39, no. 3, pp. 923–947, 2009.

[31] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang, "On graph problems in a semi-streaming model," *Theoretical Computer Science*, vol. 348, no. 2-3, pp. 207–216, 2005.

BIBLIOGRAPHY

[32] K. J. Ahn, S. Guha, and A. McGregor, "Analyzing graph structure via linear measurements," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2012, pp. 459–467.

[33] M. S. Crouch, A. McGregor, and D. Stubbs, "Dynamic graphs in the sliding-window model," in *European Symposium on Algorithms*. Springer, 2013, pp. 337–348.

[34] J. Kelner and A. Levin, "Spectral sparsification in the semi-streaming setting," in *Symposium on Theoretical Aspects of Computer Science (STACS)*, 2011.

[35] A. Goel, M. Kapralov, and I. Post, "Single pass sparsification in the streaming model with edge deletions," *arXiv preprint arXiv:1203.4900*, 2012.

[36] K. J. Ahn, S. Guha, and A. McGregor, "Spectral sparsification in dynamic graph streams," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2013, pp. 1–10.

[37] I. Abraham, D. Durfee, I. Koutis, S. Krinninger, and R. Peng, "On fully dynamic graph sparsifiers," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 335–344.

[38] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford, "Single pass spectral sparsification in dynamic streams," *SIAM Journal on Computing*, vol. 46, no. 1, pp. 456–477, 2017.

[39] I. Abraham, S. Chechik, D. Delling, A. V. Goldberg, and R. F. Werneck, "On dynamic approximate shortest paths for planar graphs with worst-case costs," in *Proceedings of*

the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). Society for Industrial and Applied Mathematics, 2016, pp. 740–753.

[40] S. Baswana, M. Gupta, and S. Sen, "Fully dynamic maximal matching in o($\log n$) update time," *SIAM Journal on Computing*, vol. 44, no. 1, pp. 88–113, 2015.

[41] A. Bernstein and C. Stein, "Faster fully dynamic matchings with small approximation ratios," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2016, pp. 692–711.

[42] S. Baswana, "Streaming algorithm for graph spanners-single pass and constant processing time per edge," *Information Processing Letters*, vol. 106, no. 3, pp. 110–114, 2008.

[43] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: sparsification, spanners, and subgraphs," in *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. ACM, 2012, pp. 5–14.

[44] S. Baswana, S. Khurana, and S. Sarkar, "Fully dynamic randomized algorithms for graph spanners," *ACM Transactions on Algorithms (TALG)*, vol. 8, no. 4, p. 35, 2012.

[45] C. Boutsidis, D. P. Woodruff, and P. Zhong, "Optimal principal component analysis in distributed and streaming models," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, ACM. https://arxiv.org/pdf/1504.06729, 2016, pp. 236–249.

[46] Z. Song, D. P. Woodruff, and P. Zhong, "Low rank approximation with entrywise ℓ_1-norm error," in *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*, ACM. https://arxiv.org/pdf/1611.00898, 2017.

[47] ——, "Relative error tensor low rank approximation," in *ArXiv preprints*. https://arXiv.org/pdf/1704.08246, 2017.

[48] A. McGregor, "Graph stream algorithms: a survey," *ACM SIGMOD Record*, vol. 43, no. 1, pp. 9–20, 2014.

[49] P. Indyk, "Algorithms for dynamic geometric problems over data streams," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM, 2004, pp. 373–380.

[50] G. Frahling and C. Sohler, "Coresets in dynamic geometric data streams," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. ACM, 2005, pp. 209–217.

[51] Z. Allen-Zhu and Y. Li, "First efficient convergence for streaming k-pca: a global, gap-free, and near-optimal rate," *arXiv preprint arXiv:1607.07837*, 2016.

[52] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of mathematical biology*, vol. 15, no. 3, pp. 267–273, 1982.

[53] L. F. Yang, V. Braverman, T. Zhao, and M. Wang, "Dynamic partition of complex networks," *arXiv preprint arXiv:1705.07881*, 2017.

[54] Z. Chen, L. F. Yang, C. J. Li, and T. Zhao, "Online multiview representation learning: Dropping convexity for better efficiency," *arXiv preprint arXiv:1702.08134*, 2017.

[55] C. J. Li, Z. Wang, and H. Liu, "Online ica: Understanding global dynamics of nonconvex optimization via diffusion processes," in *Advances in Neural Information Processing Systems*, 2016, pp. 4967–4975.

[56] V. Braverman, S. R. Chestnut, R. Krauthgamer, and L. F. Yang, "Sketches for matrix norms: Faster, smaller and more general," *arXiv preprint arXiv:1609.05885*, 2016.

[57] V. Braverman, G. Frahling, H. Lang, C. Sohler, and L. F. Yang, "Clustering high dimensional dynamic data streams," *arXiv preprint arXiv:1706.03887*, 2017.

[58] "List of open problems in sublinear algorithms: Problem 5," http://sublinear.info/5.

[59] S. Guha, P. Indyk, and A. McGregor, "Sketching information divergences," in *Learning theory*. Springer, 2007, pp. 424–438.

[60] A. Andoni, R. Krauthgamer, and I. Razenshteyn, "Sketching and embedding are equivalent for norms," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 2015, pp. 479–488.

[61] Y. Li, H. L. Nguyen, and D. P. Woodruff, "Turnstile streaming algorithms might as well be linear sketches," in *Proceedings of the 46th annual ACM Symposium on Theory of Computing*. ACM, 2014, pp. 174–183.

[62] S. Wu, F. Li, S. Mehrotra, and B. C. Ooi, "Query optimization for massively parallel

data processing," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 12.

[63] E. Jahani, M. J. Cafarella, and C. Ré, "Automatic optimization for mapreduce programs," *Proceedings of the VLDB Endowment*, vol. 4, no. 6, pp. 385–396, 2011.

[64] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with mapreduce: a survey," *ACM SIGMOD Record*, vol. 40, no. 4, pp. 11–20, 2012.

[65] S. R. Chestnut, "Stream sketches, sampling, and sabotage," Ph.D. dissertation, Johns Hopkins University, 2015, https://jscholarship.library.jhu.edu/handle/1774.2/37935.

[66] V. Braverman and R. Ostrovsky, "Generalizing the layering method of Indyk and Woodruff: Recursive sketches for frequency-based vectors on streams," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2013, pp. 58–70.

[67] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Automata, Languages and Programming*. Springer, 2002, pp. 693–703.

[68] Y. Li and D. P. Woodruff, "A tight lower bound for high frequency moment estimation with small error," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2013, pp. 623–638.

[69] I. Kremer, N. Nisan, and D. Ron, "On randomized one-round communication complexity," *Computational Complexity*, vol. 8, no. 1, pp. 21–49, 1999.

[70] A. Chakrabarti, S. Khot, and X. Sun, "Near-optimal lower bounds on the multi-party

communication complexity of set disjointness," in *18th IEEE Annual Conference on Computational Complexity*, 2003, pp. 107–117.

[71] A. Gronemeier, "Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness," in *STACS 2009: 26th International Symposium on Theoretical Aspects of Computer Science*, ser. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2009, vol. 3, pp. 505–516.

[72] T. S. Jayram, "Hellinger strikes back: A note on the multi-party information complexity of AND," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2009, pp. 562–573.

[73] Z. Bar-Yossef, "The complexity of massive data set computations," Ph.D. dissertation, University of California at Berkeley, 2002.

[74] V. Braverman, R. Ostrovsky, and A. Roytman, "Zero-one laws for sliding windows and universal sketches," in *APPROX/RANDOM 2015*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 40. Schloss Dagstuhl, 2015, pp. 573–590.

[75] "List of open problems in sublinear algorithms," http://sublinear.info/.

[76] A. Chakrabarti, K. Do Ba, and S. Muthukrishnan, "Estimating entropy and entropy norm on data streams," *Internet Mathematics*, vol. 3, no. 1, pp. 63–78, 2006.

[77] A. Chakrabarti, G. Cormode, and A. McGregor, "A near-optimal algorithm for com-

puting the entropy of a stream," in *18th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2007, pp. 328–335.

[78] N. J. Harvey, J. Nelson, and K. Onak, "Sketching and streaming entropy via approximation theory," in *49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2008, pp. 489–498.

[79] T. Jayram and D. P. Woodruff, "The data stream space complexity of cascaded norms," in *50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2009, pp. 765–774.

[80] T. Jayram, "On the information complexity of cascaded norms with small domains," in *Information Theory Workshop (ITW), 2013 IEEE*. IEEE, 2013, pp. 1–5.

[81] R. Bhatia, *Matrix analysis*, ser. Graduate Texts in Mathematics. Springer-Verlag, New York, 1997, vol. 169.

[82] V. D. Milman and G. Schechtman, *Asymptotic theory of finite-dimensional normed spaces*, ser. Lecture Notes in Mathematics. Springer-Verlag, 1986, vol. 1200.

[83] B. Klartag and R. Vershynin, "Small ball probability and Dvoretzkys theorem," *Israel Journal of Mathematics*, vol. 157, no. 1, pp. 193–207, 2007.

[84] D. M. Kane, J. Nelson, and D. P. Woodruff, "On the exact space complexity of sketching and streaming small norms," in *21st Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2010, pp. 1161–1178.

[85] A. Argyriou, R. Foygel, and N. Srebro, "Sparse prediction with the k-

support norm," in *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1457–1465. [Online]. Available: http://papers.nips.cc/paper/4537-sparse-prediction-with-the-k-support-norm.pdf

[86] A. M. McDonald, M. Pontil, and D. Stamos, "Spectral k-support norm regularization," in *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 3644–3652. [Online]. Available: http://papers.nips.cc/paper/5297-spectral-k-support-norm-regularization.pdf

[87] B. Wu, C. Ding, D. Sun, and K.-C. Toh, "On the Moreau–Yosida regularization of the vector k-norm related functions," *SIAM Journal on Optimization*, vol. 24, no. 2, pp. 766–794, 2014.

[88] S. Ganguly and G. Cormode, "On estimating frequency moments of data streams," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2007, pp. 479–493.

[89] P. Li, "Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections," in *19th Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '08. SIAM, 2008, pp. 10–19. [Online]. Available: http://dl.acm.org/citation.cfm?id=1347082.1347084

[90] A. Andoni, R. Krauthgamer, and K. Onak, "Streaming algorithms via precision sampling," in *IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 2011, pp. 363–372.

[91] V. Braverman and R. Ostrovsky, "Approximating large frequency moments with pick-and-drop sampling," in *APPROX/RANDOM 2013*, ser. Lecture Notes in Computer Science. Springer, 2013, vol. 8096, pp. 42–57.

[92] V. Braverman, J. Katzman, C. Seidell, and G. Vorsanger, "An optimal algorithm for large frequency moments using o (n1- 2/k) bits," *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pp. 4–6, 2014.

[93] S. Ganguly, "Taylor polynomial estimator for estimating frequency moments," in *Automata, Languages, and Programming*, ser. Lecture Notes in Computer Science. Springer, 2015, vol. 9134, pp. 542–553.

[94] M. Saks and X. Sun, "Space lower bounds for distance approximation in the data stream model," in *34th Annual ACM Symposium on Theory of Computing*, 2002, pp. 360–369.

[95] A. Andoni, H. L. Nguyen, Y. Polyanskiy, and Y. Wu, "Tight lower bound for linear sketches of moments," in *40th International Conference on Automata, Languages, and Programming*, ser. ICALP'13. Springer-Verlag, 2013, pp. 25–32.

[96] F. John, "Extremum problems with inequalities as subsidiary conditions," in *Studies and Essays Presented to R. Courant on his 60th Birthday.* Interscience Publishers, Inc., 1948, pp. 187–204.

[97] M. Ledoux and M. Talagrand, *Probability in Banach Spaces: isoperimetry and pro-*

cesses. Springer Science & Business Media, 2013, vol. 23.

[98] B. Laurent and P. Massart, "Adaptive estimation of a quadratic functional by model selection," *Annals of Statistics*, vol. 28, no. 5, pp. 1302–¡96¿1338, 2000.

[99] N. Nisan, "Pseudorandom generators for space-bounded computation," *Combinatorica*, vol. 12, no. 4, pp. 449–461, 1992.

[100] V. Braverman, G. Frahling, H. Lang, C. Sohler, and L. F. Yang, "Clustering high dimensional dynamic data streams," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 576–585. [Online]. Available: http://proceedings.mlr.press/v70/braverman17a.html

[101] K. Chen, "On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications," *SIAM J. Comput.*, vol. 39, no. 3, pp. 923–947, 2009. [Online]. Available: http://dx.doi.org/10.1137/070699007

[102] S. Ganguly, "Counting distinct items over update streams," in *International Symposium on Algorithms and Computation*. Springer, 2005, pp. 505–514.

[103] J. Feigenbaum, S. Kannan, and J. Zhang, "Computing diameter in the streaming and sliding-window models," *Algorithmica*, vol. 41, no. 1, pp. 25–41, 2005.

[104] P. Indyk, "Better algorithms for high-dimensional proximity problems via asymmetric embeddings," in *Proceedings of the fourteenth annual ACM-SIAM symposium on*

Discrete algorithms. Society for Industrial and Applied Mathematics, 2003, pp. 539–545.

[105] G. Cormode and S. Muthukrishnan, "Radial histograms for spatial streams," *DIM ACS Technical Report*, vol. 11, 2003.

[106] J. Hershberger and S. Suri, "Adaptive sampling for geometric problems over data streams," in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2004, pp. 252–262.

[107] T. M. Chan, "Faster core-set constructions and data stream algorithms in fixed dimensions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004, pp. 152–159.

[108] T. M. Chan and B. S. Sadjad, "Geometric optimization problems over sliding windows," *International Journal of Computational Geometry & Applications*, vol. 16, no. 02n03, pp. 145–157, 2006.

[109] A. Bagchi, A. Chaudhary, D. Eppstein, and M. T. Goodrich, "Deterministic sampling and range counting in geometric data streams," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 2, p. 16, 2007.

[110] D. Feldman, M. Monemizadeh, and C. Sohler, "A PTAS for k-means clustering based on weak coresets," in *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, 2007, pp. 11–18. [Online]. Available: http://doi.acm.org/10.1145/1247069.1247072

[111] D. Feldman, M. Schmidt, and C. 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, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, 2013, pp. 1434–1453. [Online]. Available: http://dx.doi.org/10.1137/1.9781611973105.103

[112] D. Feldman and M. Langberg, "A unified framework for approximating and clustering data," in *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, 2011, pp. 569–578. [Online]. Available: http://doi.acm.org/10.1145/1993636.1993712

[113] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental clustering and dynamic information retrieval," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997, pp. 626–635.

[114] A. Meyerson, "Online facility location," in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. IEEE, 2001, pp. 426–431.

[115] P. Indyk and E. Price, "K-median clustering, model-based compressive sensing, and sparse recovery for earth mover distance," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 2011, pp. 627–636.

[116] K. G. Larsen, J. Nelson, H. L. Nguyen, and M. Thorup, "Heavy hitters via cluster-preserving clustering," in *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, IEEE. https://arxiv.org/pdf/1604.01357, 2016, pp. 61–70.

Vita

Lin Yang was born in Guilin, Guangxi Province, China. He obtained a bachelor's degree in Math & Physics from Tsinghua University, Beijing China (2011). He earned an Ms.E degree in Computer Science from Johns Hopkins University (2015). He obtained a Ph.D. degree in Physics from Johns Hopkins University (2017).