Fast Distance Oracles for Any Symmetric Norm

Yichuan Deng
University of Science and Technology of China
ethandeng02@gmail.com

Zhao Song
Adobe Research
zsong@adobe.com

Omri Weinstein
The Hebrew University and Columbia University
ow2161@columbia.edu

Ruizhe Zhang
The University of Texas at Austin
ruizhe@utexas.edu

Abstract

In the *Distance Oracle* problem, the goal is to preprocess n vectors $x_1, x_2, \ldots, x_n$ in a d-dimensional metric space $(\mathbb{X}^d, \|\cdot\|_l)$ into a cheap data structure, so that given a query vector $q \in \mathbb{X}^d$ and a subset $S \subseteq [n]$ of the input data points, all distances $\|q - x_i\|_l$ for $x_i \in S$ can be quickly approximated (faster than the trivial $\sim d|S|$ query time). This primitive is a basic subroutine in machine learning, data mining and similarity search applications. In the case of ℓ_p norms, the problem is well understood, and optimal data structures are known for most values of p.

Our main contribution is a fast $(1 \pm \varepsilon)$ distance oracle for *any symmetric* norm $\|\cdot\|_l$. This class includes ℓ_p norms and Orlicz norms as special cases, as well as other norms used in practice, e.g. top-k norms, max-mixture and sum-mixture of ℓ_p norms, small-support norms and the box-norm. We propose a novel data structure with $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$ preprocessing time and space, and $t_q = \widetilde{O}(d + |S| \cdot \mathrm{mmc}(l)^2)$ query time, for computing distances to a subset S of data points, where $\mathrm{mmc}(l)$ is a complexity-measure (concentration modulus) of the symmetric norm. When $l = \ell_p$, this runtime matches the aforementioned state-of-art oracles.

1 Introduction

Estimating and detecting similarities in datasets is a fundamental problem in computer science, and a basic subroutine in most industry-scale ML applications, from optimization [CMF+20, CLP+20, XSS21] and reinforcement learning [SSX21], to discrepancy theory [SXZ22] and covariance estimation [Val12, Alm19], Kernel SVMs [CS09, SSSSC11], compression and clustering [IRW17, MMR22], to mention a few. Such applications often need to quickly compute distances of online (test) points to a subset of points in the input data set (e.g., the training data) for transfer-learning and classification. These applications have motivated the notion of *distance oracles* (DO) [Pel00, GPPR04, WP11]: In this problem, the goal is to preprocess a dataset of n input points $X = (x_1, x_2, \ldots, x_n)$ in some d-dimensional metric space, into a small-space data structure which, given a query vector q and a subset $\mathsf{S} \subseteq [n]$, can quickly estimate all the distances $\mathsf{d}(q, x_i)$ of q to S (note that the problem of estimating a *single* distance $\mathsf{d}(q, x_i)$ is not interesting in $\mathbb{R}^d$, as this can be trivially done in $O(d)$ time, which is necessary to merely read the query q). The most well-studied case (in both theory and practice) is when the metric space is in fact a *normed* space, i.e., the data points $\{x_i\}_{i\in[n]} \in \mathbb{R}^d$ are endowed with some predefined norm $\|\cdot\|$, and the goal is to quickly estimate $\|x_i - q\|$ simultaneously for all i, i.e., in time $\ll nd$ which is the trivial query time. Distance oracles can therefore be viewed as generalizing *matrix-vector* multiplication: for the inner-product distance function $\langle x_i, q\rangle$, the query asks to approximate $X \cdot q$ in $\ll nd$ time.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

For the most popular distance metrics—the Euclidean distance (ℓ_2-norm) and Manhattan distance (ℓ_1-norm)—classic dimension-reduction (sketching) provide very efficient distance oracles [JL84, AC09, LDFU13]. However, in many real-world problems, these metrics do not adequately capture similarities between data points, and a long line of work has demonstrated that more complex (possibly *learnable*) metrics can lead to substantially better prediction and data compression [DKJ+07]. In particular, many works over the last decade have been dedicated to extending various optimization problems beyond Euclidean/Manhattan distances, for example in (kernel) linear regression [SWY+19], approximate nearest neighbor [ANN+17, ANN+18], sampling [LSV18], matrix column subset selection [SWZ19], and statistical queries [LNRW19].

For ℓ_p norms, the DO problem is well-understood [BYJKS04], where the standard tool for constructing the data structure is via randomized *linear sketching* : The basic idea is to reduce the dimension (d) of the data points by applying some *sketching matrix* $\Phi \in \mathbb{R}^{m\times d}$ ($m \ll d$) to each data point x_i and store the sketch $\Phi x_i \in \mathbb{R}^m$. For a query point q, linearity then allows to estimate the distance from $\Phi(q - x_i)$. The seminal works of [JL84, AMS99, CCF04, TZ12] developed polylogarithmic-size sketching methods for the ℓ_2-norm, which was extended, in a long line of work, to any ℓ_p norm with $0 < p < 2$ [Ind06, KNPW11, CN20]. For $p > 2$, the sketch-size (d) becomes *polynomial*, yet still sublinear in d [SS02, BYJKS04].

In this work, we consider general *symmetric norms*, which generalize ℓ_p norms. More formally, a norm $l : \mathbb{R}^d \to \mathbb{R}$ is symmetric if, for all $x \in \mathbb{R}^d$ and every $d \times d$ 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 (for a broader introduction, see [Bha97] Chapter IV). Important special cases of symmetric norms are ℓ_p norms and *Orlicz norms* [ALS+18], which naturally arise in harmonic analysis and model data with sub-gaussian properties. Other practical examples of symmetric norms include top-k norms, max-mix of ℓ_p norms, sum-mix of ℓ_p norms, the k-support norm [AFS12] and the box-norm [MPS14].

Several recent works have studied dimension-reduction (sketching) for special cases of symmetric norms such as the Orlicz norm [BBC+17, SWY+19, ALS+18], for various numerical linear algebra primitives [SWZ19]. These sketching techniques are quite ad-hoc and are carefully tailored to the norm in question. It is therefore natural to ask whether *symmetry alone* is enough to guarantee dimensionality-reduction for symmetric similarity search, in other words:

Is there an efficient $(1 + \epsilon)$-distance oracle for general symmetric norms?

By "efficient", we mean small space and preprocessing time (ideally $\sim nd$), fast query time ($\ll d|S|$ for a query $(q, S \subseteq [n])$) and ideally supporting dynamic updates to x_i's in $\widetilde{O}(d)$ time. Indeed, most ML applications involve rapidly-changing *dynamic datasets*, and it is becoming increasingly clear that static data structures do not adequately capture the requirements of real-world applications [JKDG08, CMF+20, CLP+20]. As such, it is desirable to design a *dynamic distance oracle* which has both small update time (t_u) for adding/removing a point $x_i \in \mathbb{R}^d$, and small query time (t_q) for distance estimation. We remark that most known DOs are dynamic by nature (as they rely on linear sketching techniques), but for general metrics (e.g., graph shortest-path or the ℓ_∞ norm) this is much less obvious, and indeed *linearity* of encoding/decoding will be a key challenge in our data structure (see next section). The problem is formally defined as follows:

Definition 1.1 (Symmetric-Norm Distance Oracles)**.** *Let $\|\cdot\|_{\text{sym}}$ be the symmetric norm. The* Symmetric-norm Distance Oracle *is a data structure that efficiently supports any sequence of the following operations:*

- INIT$(\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d, \epsilon \in (0, 1), \delta \in (0, 1))$. *The data structure takes n data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter ϵ and a failure probability δ as input.*
- UPDATEX$(z \in \mathbb{R}^d, i \in [n])$. *Update the data structure with the i-th new data point z.*
- ESTPAIR$(i, j \in [n])$ *Outputs a number* pair *such that* $(1 - \epsilon)\|x_i - x_j\|_{\text{sym}} \leq \text{pair} \leq (1 + \epsilon) \cdot \|x_i - x_j\|_{\text{sym}}$ *with probability at least* $1 - \delta$.
- QUERY$(q \in \mathbb{R}^d)$. *Outputs a vector* $\text{dst} \in \mathbb{R}^n$ *such that* $\forall i \in [n], (1 - \epsilon)\|q - x_i\|_{\text{sym}} \leq \text{dst}_i \leq (1 + \epsilon)\|q - x_i\|_{\text{sym}}$. *with probability at least* $1 - \delta$.

where $\|x\|_{\text{sym}}$ is the symmetric norm of the vector x.

This problem can be viewed as an *online* version of the (approximate) *closest-pair* problem [Val12], which asks to find the closest pair of points among an *offline batch* of data points $X = x_1, \ldots, x_n \in \mathbb{R}^d$, or equivalently, the smallest entry of the *covariance matrix* $XX^\top$. One major (theoretical) advantage of the offline case is that it enables the use of fast matrix-multiplication (FMM) to speed-up the computation of the covariance matrix [Val12, AWY14, AWY18, Alm19] (i.e., sub-linear amortized per query). By contrast, in the online setting such speedups are conjectured to be impossible [HKNS15, LW17].

Notations. For any positive integer n, we use $[n]$ to denote $\{1, 2, \ldots, n\}$. For any function f, we use $\widetilde{O}(f)$ to denote $f \cdot \mathrm{poly}(\log f)$. We use $\Pr[\cdot]$ to denote probability. We use $\mathbb{E}[\cdot]$ to denote expectation. We use both $l(\cdot)$ and $\|\cdot\|_{\mathrm{sym}}$ to denote the symmetric norm. We use $\|\cdot\|_2$ to denote the entry-wise ℓ_2 norm. We define a tail notation which is very standard in sparse recover/compressed sensing literature. For any given vector $x \in \mathbb{R}^d$ and an integer k, we use $x_{\overline{[k]}}$ or $x_{\mathrm{tail}(k)}$ to denote the vector that without (zeroing out) top-k largest entries (in absolute). For a vector x, we use $x^\top$ to denote the transpose of x. For a matrix A, we use $A^\top$ to denote the transpose of A. We use $\mathbf{1}_n$ denote a length-n vector where every entry 1. We use $\mathbf{0}_n$ denote a length-n vector where every entry is 0.

1.1 Our Results

Two important complexity measures of (symmetric) norms, which capture their "intrinsic dimensionality", are the *concentration modulus* (mc) and *maximum modulus* (mmc) parameters. We now define these quantities along the lines of [BBC+17, SWY+19].

Definition 1.2 (Modulus of concentration (mc))**.** *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 $\Pr[l(X) \geq M_l] \geq 1/2$ and $\Pr[l(X) \leq M_l] \geq 1/2$. Similarly, $\mathfrak{b}_l$ denotes the maximum value of $l(x)$ over $x \in S^{n-1}$. We call the ratio*

$$\mathrm{mc}(l) := \mathfrak{b}_l / M_l$$

the modulus of concentration of the norm l.

For every $k \in [n]$, the norm l induces a norm $l^{(k)}$ on $\mathbb{R}^k$ by setting

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

where $(x_1, x_2, \ldots, x_k, 0, \ldots, 0) \in \mathbb{R}^n$.

Definition 1.3 (Maximum modulus of concentration (mmc))**.** *Define the maximum modulus of concentration of the norm l as*

$$\mathrm{mmc}(l) := \max_{k \in [n]} \mathrm{mc}(l^{(k)}) = \max_{k \in [n]} \frac{\mathfrak{b}_{l^{(k)}}}{M_{l^{(k)}}}$$

Next, we present a few examples (in Table 1) for different norm l's $\mathrm{mmc}(l)$.

Norm l	$\mathrm{mmc}(l)$
$\ell_p (p \leq 2)$	$\Theta(1)$
$\ell_p (p > 2)$	$\Theta(d^{1/2-1/p})$
top-k norms	$\widetilde{\Theta}(\sqrt{d/k})$
k-support norms and the box-norm	$O(\log d)$
max-mix and sum-mix of ℓ_1 and ℓ_2	$O(1)$
Orlicz norm $\|\cdot\|_G$	$O(\sqrt{C_G \log d})$

Table 1: Examples of $\mathrm{mmc}(l)$, where max-mix of ℓ_1 and ℓ_2 is defined as $\max\{\|x\|_2, c\|x\|_1\}$ for a real number c, sum-mix of ℓ_1 and ℓ_2 is defined as $\|x\|_2 + c\|x\|_1$ for a real number c, and C_G of Orlicz norm is defined as the number that for all $0 < x < y$, $G(y)/G(x) \leq C_G (y/x)^2$.

We are now ready to state our main result:

Theorem 1.4 (Main result, informal version of Theorem D.1)**.** *Let $\|\cdot\|_l$ be any symmetric norm on $\mathbb{R}^d$. There is a data structure for the online symmetric-norm Distance Oracle problem (Definition 1.1), which uses $n(d + \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))$ space, supporting the following operations:*

- $\textsc{Init}(\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d, \epsilon \in (0,1), \delta \in (0,1))$*: Given n data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter ϵ and a failure probability δ as input, the data structure preprocesses in time*

  $$n(d + \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta)).$$

  Note that $\mathrm{mmc}()$ *is defined as Definition 1.3.*

- $\textsc{UpdateX}(z \in \mathbb{R}^d, i \in [n])$*: Given an update vector $z \in \mathbb{R}^d$ and index $i \in [n]$, the data structure receives z and i as inputs, and updates the i-th data point $x_i \leftarrow z$, in*

  $$d \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))$$

  time.

- $\textsc{Query}(q \in \mathbb{R}^d, \mathsf{S} \subseteq [n])$*: Given a query point $q \in \mathbb{R}^d$ and a subset of the input points $\mathsf{S} \subseteq [n]$, the* $\textsc{Query}$ *operation outputs a $(1+\epsilon)$- multiplicative approximation to each distance from q to points in S, in time*

  $$(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))$$

  i.e. it provides a set of estimates $\{\mathrm{dst}_i\}_{i \in \mathsf{S}}$ such that:

  $$\forall i \in \mathsf{S}, (1-\epsilon)\|q - x_i\|_l \leq \mathrm{dst}_i \leq (1+\epsilon)\|q - x_i\|_l$$

  with probability at least $1 - \delta$.

- $\textsc{EstPair}(i, j \in [n])$ *Given indices $i, j \in [n]$, the* $\textsc{EstPair}$ *operation takes i and j as input and approximately estimates the symmetric norm distances from i-th to the j-th point $x_i, x_j \in \mathbb{R}^d$ in time $\mathrm{mmc}(l)^2 \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))$ i.e., it provides an estimated distance* pair *such that:*

  $$(1-\epsilon)\|x_i - x_j\|_l \leq \mathrm{pair} \leq (1+\epsilon)\|x_i - x_j\|_l$$

  with probability at least $1 - \delta$.

Roadmap. In Section 2, we give an overview of the techniques that we mainly use in the work. In Section 3 we give an introduction of the Sparse Recovery Data we use for sketching. In Section 4 we analyze the running time and space of our data structure with their proofs, respectively. In Section 5 we show the correctness of our data structure and give its proof. Finally in Section 6 we conclude our work.

2 Technique Overview

Our distance oracle follows the "sketch-and-decode" approach, which was extensively used in many other sublinear-time compressed sensing and sparse recovery problems [Pri11, HIKP12, LNNT16, NS19, SSWZ22]. The main idea is to compress the data points into smaller dimension by computing, for each data point $x_i \in \mathbb{R}^d$, a (randomized) linear sketch $\Phi \cdot x_i \in \mathbb{R}^{d'}$ with $d' \ll d$ at preprocessing time, where Φx_i is an unbiased estimator of $\|x_i\|$. At query time, given a query point $q \in \mathbb{R}^d$, we analogously compute its sketch Φq. By *linearity* of Φ, the distance between q and x_i (i.e., $\ell(q - x_i)$) can be trivially decoded from the sketch *difference* $\Phi q - \Phi x_i$. As we shall see, this simple virtue of linearity is less obvious to retain when dealing with general symmetric norms.

Layer approximation Our algorithm uses the *layer approximation* method proposed by Indyk and Woodruff in [IW05] and generalized to symmetric norms in [BBC+17, SWY+19]. Since symmetric norms are invariant under reordering of the coordinates, the main idea in [IW05] is to construct a "layer vector" as follows: for a vector $v \in \mathbb{R}^d$, round (the absolute value of) each coordinate to the nearest power α^i for some fixed $\alpha \in \mathbb{R}$ and $i \in \mathbb{N}$, and then sort the coordinates in an increasing order.

This ensures that the i-th layer contains all the coordinates $j \in [d]$ satisfying: $\alpha^{i-1} < |v_j| \leq \alpha^i$. In particular, the layer vector of v has the form

$$\mathcal{L}(v) := (\underbrace{\alpha^1, \ldots, \alpha^1}_{b_1 \text{ times}}, \underbrace{\alpha^2, \ldots, \alpha^2}_{b_2 \text{ times}}, \cdots, \underbrace{\alpha^P, \ldots, \alpha^P}_{b_P \text{ times}}, 0, \ldots, 0) \in \mathbb{R}^d,$$

where b_i is the number of coordinates in layer-i. More importantly, since the norm is symmetric, the layer vector $\mathcal{L}(v)$ has a succinct representation: $(b_i)_{i \in [P]}$.

Then, it suffices to estimate b_i for each $i \in [P]$, where the Indyk-Woodruff sketching technique can be used to approximate the vector. At the i-th layer, each coordinate of v is sampled with probability P/b_i, and then the algorithm identifies the ℓ_2-*heavy-hitters* of the sampled vector. [BBC+17] gave a criterion for identifying the important layers, whose heavy-hitter coordinates in the corresponding *sampled* vectors, is enough to recover the entire symmetric norm $\|v\|_{\text{sym}}$.

Unfortunately, this technique for norm estimation does not readily translate to estimating *distances* efficiently:

- *Too many layers:* In previous works, each data point x_i is sub-sampled *independently* in R layers, i.e, generates R subsets of coordinates $S_i^1, \ldots, S_i^R \subset [d]$. The sketch of the query point $S(q)$ then needs to be compared to each $S(x_i)$ in every layer. Since there are $R = \Omega(n)$ layers in [BBC+17] of size $\Omega(d)$ across all data points, the total time complexity will be at least $\Omega(nd)$, which is the trivial query time.
- *Non-linearity:* The aforementioned sketching algorithms [BBC+17] involve nonlinear operations, and thus cannot be directly used for distance estimation.
- *Slow decoding:* The aforementioned sketches take linear time to decode the distance from the sketch, which is too slow for our application.

To overcome these challenges, we use the following ideas:

Technique I: shared randomness To reduce the number of layers, we let all the data points use the same set of layers. That is, in the initialization of our algorithm, we independently sample R subsets $S^1, \ldots, S^R$ with different probabilities. Then, for each data point x_i, we consider $(x_i)_{S^j}$ as the sub-sample for the j-th layer, and perform sketch on it. Hence, the number of different layer sets is reduced from nt to t, where t is the number of layers for each data point. We share the layers for all points to remove the n factor. For a query point q, we just need to compute the sketches for $(q)_{S^1}, \ldots, (q)_{S^R}$. And the distance between q and x_i can be decoded from $\{\Phi(x_i)_{S^j} - \Phi(q)_{S^j}\}_{j \in [R]}$. We also prove that the shared randomness in all data points will not affect the correctness of layer approximation.

Technique II: linearization We choose a different sketching method called BATCHHEAVYHITTER (see Theorem D.3 for details) to generate and maintain the *linear sketches*, which allows us to decode the distance from sketch difference.

Technical III: locate-and-verify decoding We design a *locate-and-verify* style decoding method to recover distance from sketch. In our data structure, we not only store the sketch of each sub-sample vector, but also the vector itself. Then, in decoding a sketch, we can first apply the efficient sparse-recovery algorithm to identify the position of heavy-hitters. Next, we directly check the entries at those positions to verify that they are indeed "heavy" (comparing the values with some threshold), and drop the non-heavy indices. This verification step is a significant difference from the typical sparse recovery approaches, which employ complex (and time-consuming) subroutines to *reconstruct the values* of the heavy-hitter coordinates. Instead, our simple verification procedure eliminates all the *false-positive* heavy-hitters, therefore dramatically reducing the running time of the second step, which can now be performed directly by reading-off the values from the memory.

With these three techniques, we obtain our sublinear-time distance estimation algorithm. Our data structure first generate a bunch of randomly selected subsets of coordinates as the layer sets. Then, for each data point, we run the BATCHHEAVYHITTER procedure to sketch the sub-sample vector in each layer[1]. In the query phase, we call the DECODE procedure of BATCHHEAVYHITTER for the

[1]The total sketch size of each data point is $\mathrm{mmc}(l)^2 \cdot \mathrm{poly}\log d$. In the ℓ_p-norm case with $p > 2$, $\mathrm{mmc}(l) = d^{1/2-1/p}$ (see Table 1) and our sketch size is $\widetilde{O}(d^{1-2/p})$, matching the lower bound of ℓ_p-sketching. When $p \in (0, 2]$, $\mathrm{mmc}(l) = \Theta(1)$ and our sketch size is $\widetilde{O}(1)$, which is also optimal.

sketch differences between the query point q and each data point x_i, and obtain the heavy hitters of each layer. We then select some "important layers" and use them to approximately recover the layer vector $\mathcal{L}(q - x_i)$, which gives the estimated distance $\|q - x_i\|_{\mathrm{sym}}$.

Finally, we summarize the time and space costs of our data structure. Let ϵ be the precision parameter and δ be the failure probability parameter. Our data structure achieves $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$-time for initialization , $\widetilde{O}(d)$-time per data point update, and $\widetilde{O}(d + n \cdot \mathrm{mmc}(l)^2))$-time per query. As for space cost, our data structure uses the space of $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$ in total. Note that mmc is defined as Defnition 1.3.

3 Sparse Recovery Data Structure

We design a data structure named BATCHHEAVYHITTER to generate sketches and manage them. In our design, it is a "linear sketch" data structure, and providing the following functions:

- INIT($\epsilon \in (0, 0.1), n, d$). Create a set of Random Hash functions and all the n copies of sketches share the same hash functions. This step takes $\mathcal{T}_{\mathrm{init}}(\epsilon, n, d)$ time.
- ENCODE($i \in [n], z \in \mathbb{R}^d, d$). This step takes $\mathcal{T}_{\mathrm{encode}}(d)$ encodes z into i-th sketched location and store a size $\mathcal{S}_{\mathrm{space}}$ linear sketch.
- ENCODESINGLE($i \in [n], j \in [d], z \in \mathbb{R}, d$). This step takes $\mathcal{T}_{\mathrm{encodesingle}}(d)$ updates one sparse vector $e_j z \in \mathbb{R}^d$ into i-th sketched location.
- SUBTRACT($i, j, l \in [n]$). Update the sketch at i-th location by j-th sketch minus l-th sketch.
- DECODE($i \in [n], \epsilon \in (0, 0.1), d$). This step takes $\mathcal{T}_{\mathrm{decode}}(\epsilon, d)$ such that it returns a set $L \subseteq [d]$ of size $|L| = O(\epsilon^{-2})$ containing all ϵ-heavy hitters $i \in [n]$ under ℓ_p. Here we say i is an ϵ-heavy hitter under ℓ_2 if $|x_i| \geq \epsilon \cdot \|x_{\overline{[\epsilon^{-2}]}}\|_2$ where $x_{\overline{[k]}}$ denotes the vector x with the largest k entries (in absolute value) set to zero. Note that the number of heavy hitters never exceeds $2/\epsilon^2$.

With this data structure, we are able to generate the sketches for each point, and subtract each other with its function. And one can get the output of heavy hitters of each sketch stored in it with DECODE function. More details are deferred to Section D.2.

4 Running Time and Space of Our Algorithm

We first analyze the running time of different procedures of our data structure DISTANCEONSYMMETRICNORM. See Algorithm 5, with the linear sketch technique, we spend the time of $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$ for preprocessing and generate the sketches stored in the data structure. When updating the data with Algorithm 6, we spend $\widetilde{O}(d)$ to update the sketch. And when a query comes (Algorithm 7), we spend $\widetilde{O}(d + n \cdot \mathrm{mmc}(l)^2))$ to get the output distance estimation. The lemmas of running time and their proof are shown below in this section.

Lemma 4.1 (INIT time, informal)**.** *Given data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter $\epsilon \in (0, 1)$, and a failure probability $\delta \in (0, 1)$ as input, the procedure* INIT *(Algorithm 5) runs in time*

$$O(n(d + \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))).$$

Proof. The INIT time includes these parts:

- Line 12 takes $O(RLU \cdot \mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d))$ to initialize sketches
- Line 17 to Line 19 take $O(RUdL)$ to generate the bmap;
- Line 24 takes $O(ndRUL \cdot \mathcal{T}_{\mathrm{encodesingle}}(d))$ to generate sketches

By Theorems D.3, we have

- $\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d)) = n \cdot O(\beta^{-1} \log^2 d) = O(n \cdot \mathrm{mmc}(l)^2 \log^7(d)\epsilon^{-5})$,

Algorithm 1 Data structure for symmetric norm estimation: members, init, informal version of Algorithm 4 and Algorithm 5

data structure DISTANCEONSYMMETRICNORM ▷ Theorem D.1
members
  $\{x_i\}_{i=1}^n \in \mathbb{R}^d$
  $\{S_{r,l,u}\}_{r\in[R],l\in[L],u\in[U]}$ ▷ A list of the BATCHHEAVYHITTER
  $\{H_{r,l,u}\}_{r\in[R],l\in[L],u\in[U]} \subset [d] \times \mathbb{R}$
end members

public:
procedure INIT($\{x_1, \cdots, x_n\} \subset \mathbb{R}^d, \delta, \epsilon$) ▷ Lemma 4.1
  Initialize the sparse-recovery data structure $\{S_{r,l,u}\}$
  Create $\{\mathrm{bmap}_{r,l,u}\}$ shared by all $i \in [n]$ ▷ $\{\mathrm{bmap}_{r,l,u}\}$ is list of a layer set map
  for $i \in [n], j \in [d]$ **do**
    if $\mathrm{bmap}_{r,u,l}[j] = 1$ **then**
      Sample $x_{i,j}$ into each subvector $\overline{x}_{r,u,l,i}$
    end if
  end for
  Encode $\{x_{r,u,l,i}\}_{i\in[n]}$ into $\{S_{r,l,u}\}$
end procedure
end data structure

Algorithm 2 Data structure for symmetric norm estimation: query, informal version of Algorithm 7

data structure DISTANCEONSYMMETRICNORM
public:
procedure QUERY($q \in \mathbb{R}^d$) ▷ Lemma 4.3, 5.1
  Encode q into $\{S_{r,l,u}\}$
  for $i \in [n]$ **do**
    Subtract the sketch of x_i and q, get the estimated heavy-hitters of $q - x_i$
    Decode the sketch and store returned estimation of heavy-hitters into $H_{r,l,u}$
    for $u \in [U]$ **do**
      if $H_{r,l,u}$ provide correct indices of heavy hitters **then**
        Select it as good set and store it in $H_{r,l}$
      end if
    end for
    Generate estimated layer sizes $\{c_k^i\}_{k\in[P]}$
    $\mathrm{dst}_i \leftarrow$ LAYERVETCORAPPROX($\alpha, c_1^i, c_2^i, \ldots, c_P^i, d$)
    Reset $\{H_{r,l,u}\}$ for next distance
  end for
  return $\{\mathrm{dst}_i\}_{i\in[n]}$
end procedure
end data structure

- $\mathcal{T}_{\mathrm{encodesingle}}(d) = O(\log^2(d))$.

Adding them together we got the time of

$$
\begin{aligned}
& O(RLU\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d)) + O(RUdL) + O(ndRUL \cdot \mathcal{T}_{\mathrm{encodesingle}}(d)) \\
= & O(RLU(\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d) + nd \cdot \mathcal{T}_{\mathrm{encodesingle}}(d))) \\
= & O(\epsilon^{-4}\log(1/\delta)\log^4(d) \cdot \log(d) \cdot \log(d^2/\delta \cdot \log(nd))(n \cdot \mathrm{mmc}(l)^2 \log^7(d)\epsilon^{-5} + nd\log^2(d))) \\
= & O(n(d + \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))),
\end{aligned}
$$

where the first step follows from merging the terms, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encodesingle}}(d), \mathcal{T}_{\mathrm{init}}$, the third step follows from merging the terms.

Thus, we complete the proof. □

Lemma 4.2 (UPDATE time, informal)**.** *Given a new data point $z \in \mathbb{R}^d$, and an index i where it should replace the original data point $x_i \in \mathbb{R}^d$. The procedure* UPDATE *(Algorithm 6) runs in time*

$$O(d \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))$$

Proof. The UPDATE operation calls BATCHHEAVYHITTER.ENCODE for RLU times, so it has the time of

$$\begin{aligned} O(RLU \cdot \mathcal{T}_{\mathrm{encode}}(d)) &= O(\epsilon^{-4} \log(1/\delta) \log^4(d) \cdot \log(d) \cdot \log(d^2/\delta) \cdot d \log^2(d) \cdot \log(nd)) \\ &= O(\epsilon^{-4} d \log^9(nd/\delta)) \end{aligned}$$

where the first step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}(d)$, the second step follows from

$$\begin{aligned} & \log(1/\delta) \log^4(d) \log(d) \log(d^2/\delta) \log^2(d) \log(nd) \\ = & (\log(1/\delta))(\log^7 d)(2 \log d + \log(1/\delta)) \log(nd) \\ = & O(\log^9(nd/\delta)). \end{aligned}$$

Thus, we complete the proof. □

Here, we present a QUERY for outputting all the n distances. In Section E, we provide a more general version, which can take any input set $\mathsf{S} \subseteq [n]$, and output distance for only them in a shorter time that proportional to $|\mathsf{S}|$.

Lemma 4.3 (QUERY time, informal)**.** *Given a query point $q \in \mathbb{R}^d$, the procedure* QUERY *(Algorithm 7) runs in time*

$$O((d + n \cdot \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))).$$

Proof. The QUERY operation has the following two parts:

- **Part 1:** Line 5 takes $O(RLU \cdot \mathcal{T}_{\mathrm{encode}})$ time to call ENCODE to generate sketches for q.
- **Part 2:** For every $i \in [n]$:
  - Line 13 takes $O(RLU \cdot \mathcal{T}_{\mathrm{subtract}})$ time to compute sketch of the difference between q and x_i, and store the sketch at index of $n + 1$.
  - Line 14 takes $O(RLU \cdot \mathcal{T}_{\mathrm{decode}})$ time to decode the BATCHHEAVYHITTER and get estimated heavy hitters of $q - x_i$.
  - Line 16 to Line 24 takes $O(RLU \cdot 2/\beta)$ time to analyze the BATCHHEAVYHITTER and get the set of indices, where $2/\beta$ is the size of the set.
  - Line 30 takes $O(LP \cdot 2/\beta)$ time to compute size of the layer sets cut by α.
  - Line 32 to Line 36 takes $O(PL)$ time to compute the estimation of each layer.

  The total running time of this part is:

  $$\begin{aligned} & n \cdot (O(RLU \cdot \mathcal{T}_{\mathrm{subtract}}) + O(RLU \cdot \mathcal{T}_{\mathrm{decode}}) + O(RLU \cdot 2/\beta) + O(LP \cdot 2/\beta) + O(LP)) \\ = & O(nL(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1})) \end{aligned}$$

  time in total.

Taking these two parts together we have the total running time of the QUERY procedure:

$$\begin{aligned} & O(RLU \cdot \mathcal{T}_{\mathrm{encode}}) + O(nL(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1})) \\ = & O(RLU(\mathcal{T}_{\mathrm{encode}} + n \cdot \mathcal{T}_{\mathrm{subtract}} + n \cdot \mathcal{T}_{\mathrm{decode}} + n\beta^{-1}) + nLP\beta^{-1}) \\ = & O(\epsilon^{-4} \log^6(d/\delta) \log(nd)(d \log^2(d) + n\beta^{-1} \log^2(d)) + n\epsilon^{-1} \log^2(d)\beta^{-1}) \\ = & O((d + n \cdot \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(nd/\delta))) \end{aligned}$$

where the first step follows from the property of big O notation, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{subtract}}, \mathcal{T}_{\mathrm{decode}}$ (Theorem D.3) , P, the third step follows from merging the terms.

Thus, we complete the proof.

□

Next, we analyze the space usage in our algorithm. We sketch the proof in below and delay the full proof into Section G.

Lemma 4.4 (Space complexity of our data structure, informal version of Lemma G.1)**.** *Our data structure (Algorithm 1 and 2) uses space*

$$O(n(d + \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(d/\delta))).$$

Proof Sketch. First, we store the original data with space for $x = O(nd)$.

Second, we store the sub stream/sample of original data with

$$\text{space for } \overline{x} = \widetilde{O}(\epsilon^{-4} nd).$$

Third, our data structure holds a set $\{S_{r,l,u}\}_{r \in [R], l \in [L], u \in [U]}$, each $S_{r,l,u}$ of size of $\widetilde{O}(n \cdot \beta^{-1})$. We show that total space needed is

$$\text{space for } S = \widetilde{O}(\epsilon^{-9} n \cdot \mathrm{mmc}(l)^2).$$

Forth, we hold a bit-map bmap of size $O(RLUd)$, which uses the space of

$$\text{space for bmap} = \widetilde{O}(\epsilon^{-4} d).$$

Fifth, in QUERY procedure, we generate a set of sets $\{H_{r,l,u}\}_{r \in [R], l \in [L], u \in [U]}$, each of the sets has size of $O(\beta^{-1})$, so the whole set uses space of

$$\text{space for } H = \widetilde{O}(\epsilon^{-9} \cdot \mathrm{mmc}(l)^2).$$

By putting them together, the total space complexity is

$$\begin{aligned} & \text{space for } x + \text{space for } \overline{x} + \text{space for } S + \text{space for bmap} + \text{space for } H \\ = & O(nd) + \widetilde{O}(\epsilon^{-4} nd) + \widetilde{O}(\epsilon^{-9} n \cdot \mathrm{mmc}(l)^2) + \widetilde{O}(\epsilon^{-4} d) + \widetilde{O}(\epsilon^{-9} \cdot \mathrm{mmc}(l)^2) \\ = & \widetilde{O}(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2)). \end{aligned}$$

□

5 Correctness of Our Algorithm

The correctness of our distance oracle is proved in the following lemma:

Lemma 5.1 (QUERY correctness)**.** *Given a query point $q \in \mathbb{R}^d$, the procedure* QUERY *(Algorithm 7) takes q as input and approximately estimates $\{\mathrm{dst}_i\}_{i \in [n]}$ the distance between q and every x_i with the norm l, such that for every dst_i, with probability at least $1 - \delta$, we have*

$$(1 - \epsilon) \cdot \|q - x_i\|_{\mathrm{sym}} \leq \mathrm{dst}_i \leq (1 + \epsilon) \cdot \|q - x_i\|_{\mathrm{sym}}$$

Proof. Without loss of generality, we can consider a fixed $i \in [n]$. For simplicity, we denote x_i by x.

Let $v := q - x$. By Lemma C.3, it is approximated by its layer vector $\mathcal{L}(v)$, namely,

$$\|v\|_{\mathrm{sym}} \leq \|\mathcal{L}(v)\|_{\mathrm{sym}} \leq (1 + O(\epsilon))\|v\|_{\mathrm{sym}}, \tag{1}$$

where $\| \cdot \|_{\mathrm{sym}}$ is a symmetric norm, denoted also by $l(\cdot)$.

We assume without loss of generality that $\epsilon \geq 1/\mathrm{poly}(d)$. Our algorithm maintains a data structure that eventually produces a vector $\mathcal{J}(v)$, which is created with the layer sizes $c_1, c_2, \ldots, c_P$, where c's denotes the estimated layer sizes output by out data structure, and the b's are the ground truth layer sizes. We will show that with high probability, $\|\mathcal{J}(v)\|_{\mathrm{sym}}$ approximates $\|\mathcal{L}(v)\|_{\mathrm{sym}}$. Specifically, to achieve $(1 \pm \epsilon)$-approximation to $\|v\|_{\mathrm{sym}}$, we set the approximation guarantee of the layer sets (Definition C.1) to be $\epsilon_1 := O(\frac{\epsilon^2}{\log(d)})$ and the importance guarantee to be $\beta_0 := O(\frac{\epsilon^5}{\mathrm{mmc}(l)^2 \log^5(d)})$, where $\mathrm{mmc}(l)$ is defined as Definition 1.3.

Observe that the number of non-empty layer sets $P = O(\log_\alpha(d)) = O(\log(d)/\epsilon)$. Let E_{succeed} denote the event $(1-\epsilon_1)b_k \leq c_k \leq b_k$. By Lemma F.1, it happens with high probability $1-\delta$. Conditioned on this event.

Denote by $\mathcal{J}(v)$ the vector generated with the layer sizes given by Line 35, and by $\mathcal{L}^*(v)$ the vector $\mathcal{L}(v)$ after removing all buckets (Definition C.2) that are not β-contributing (Definition C.5), and define $\mathcal{J}^*(v)$ similar to $\mathcal{L}^*(v)$, where we set $\beta := \epsilon/P = O(\epsilon^2/\log(d))$. Every β-contributing layer is necessarily β_0-important (Definition C.1) by Lemma C.7 and Lemma C.8 and therefore satisfies $c_k \geq (1-\epsilon_1)b_k$ (Lemma F.1). We bound the error of $\|\mathcal{L}^*(v)\|_{\text{sym}}$ by Lemma C.6, namely,

$$(1-O(\epsilon))\|\mathcal{L}(v)\|_{\text{sym}} \leq (1-O(\log_\alpha d)\cdot\beta)\|\mathcal{L}(v)\|_{\text{sym}} \leq \|\mathcal{L}^*(v)\|_{\text{sym}} \leq \|\mathcal{L}(v)\|_{\text{sym}}.$$

where the first step follows from the definition of β, the second step and the third step follow from Lemma C.6.

Then, we have

$$\begin{aligned}
\|\mathcal{J}(v)\|_{\text{sym}} &\geq \|\mathcal{J}^*(v)\|_{\text{sym}} \\
&= \|\mathcal{L}^*(v)\backslash\mathcal{L}_{k_1}(v)\cup\mathcal{J}_{k_1}(v)\ldots\backslash\mathcal{L}_{k_\kappa}(v)\cup\mathcal{J}_{k_\kappa}(v)\|_{\text{sym}} \\
&\geq (1-\epsilon_1)^P\|\mathcal{L}^*(v)\|_{\text{sym}} \\
&\geq (1-O(\epsilon))\|\mathcal{L}^*(v)\|_{\text{sym}}. \qquad (2)
\end{aligned}$$

where the first step follows from monotonicity (Lemma B.1), the second step follows from the definition of $\mathcal{J}^*(v)$, the third step follows from Lemma C.4, and the fourth step follows from the definition of ϵ_1 and P.

Combining Eq. (1) and (2), we have

$$(1-O(\epsilon))\cdot\|v\|_{\text{sym}} \leq \|\mathcal{J}^*(v)\|_{\text{sym}} \leq \|v\|_{\text{sym}}, \qquad (3)$$

which bounds the error of $\|\mathcal{J}^*(v)\|_{\text{sym}}$ as required. Note that, with Lemma C.6 we have

$$(1-O(\epsilon))\cdot\|\mathcal{J}(v)\|_{\text{sym}} \leq \|\mathcal{J}^*(v)\|_{\text{sym}} \leq \|\mathcal{J}(v)\|_{\text{sym}} \qquad (4)$$

Combining the Eq.(3) and (4) we have

$$(1-O(\epsilon))\cdot\|v\|_{\text{sym}} \leq \|\mathcal{J}(v)\|_{\text{sym}} \leq (1+O(\epsilon))\cdot\|v\|_{\text{sym}}$$

Note that E_{succeed} has a failure probability of δ. Thus, we complete the proof. □

6 Conclusion

Similarity search is the backbone of many large-scale applications in machine-learning, optimization, databases and computational geometry. Our work strengthens and unifies a long line of work on metric embeddings and sketching, by presenting the first Distance Oracle *for any symmetric norm*, with nearly-optimal query and update times. The generality of our data structure allows to apply it as a *black-box* for data-driven *learned* symmetric distance metrics [DKJ+07] and in various optimization problems involving symmetric distances.

Our work raises several open questions for future study:

- The efficiency of our data structure depends on $\text{mmc}(l)$, the concentration property of the symmetric norm. Is this dependence necessary?
- Can we generalize our data structure to certain *asymmetric norms* ?
- Can we extend our symmetric-norm distance oracle to any (non-linear) metric space (i.e., graphs)?

We believe our work is also likely to influence other fundamental problems in high-dimensional optimization and search, e.g, kernel linear regression, geometric sampling and near-neighbor search.

Acknowledgments

RZ was supported by the University Graduate Continuing Fellowship from UT Austin.

References

[AC09] Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[AFS12] Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse prediction with the k-support norm. *Advances in Neural Information Processing Systems (NeurIPS)*, 25, 2012.

[Alm19] Josh Alman. An illuminating algorithm for the light bulb problem. In *2nd Symposium on Simplicity in Algorithms (SOSA)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[ALS+18] Alexandr Andoni, Chengyu Lin, Ying Sheng, Peilin Zhong, and Ruiqi Zhong. Subspace embedding and linear regression with orlicz norm. In *International Conference on Machine Learning (ICML)*, pages 224–233. PMLR, 2018.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

[ANN+17] Alexandr Andoni, Huy L Nguyen, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 902–913, 2017.

[ANN+18] Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 159–169. IEEE, 2018.

[AWY14] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 218–230. SIAM, 2014.

[AWY18] Josh Alman, Joshua R Wang, and Huacheng Yu. Cell-probe lower bounds from online communication complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1003–1012, 2018.

[BBC+17] Jarosław Błasiok, Vladimir Braverman, Stephen R Chestnut, Robert Krauthgamer, and Lin F Yang. Streaming symmetric norms via measure concentration. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 716–729, 2017.

[Bha97] Rajendra Bhatia. Matrix analysis, volume 169 of. *Graduate texts in mathematics*, 1997.

[BYJKS04] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[CCF04] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[CLP+20] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. Mongoose: A learnable lsh framework for efficient neural network training. In *International Conference on Learning Representations (ICLR)*, 2020.

[CMF+20] Beidi Chen, Tharun Medini, James Farwell, Charlie Tai, Anshumali Shrivastava, et al. Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *Proceedings of Machine Learning and Systems*, 2:291–306, 2020.

[CN20] Yeshwanth Cherapanamjeri and Jelani Nelson. On adaptive distance estimation. In *Advances in Neural Information Processing Systems*, 2020.

[CS09] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

[DKJ+07] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[GMS86] M Gromov, V Milman, and G Schechtman. Asymptotic theory of finite dimensional normed spaces, volume 1200 of lectures notes in mathematics, 1986.

[GPPR04] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.

[HIKP12] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 563–578, 2012.

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2015.

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

[IRW17] Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Practical data-dependent metric compression with provable guarantees. *Advances in Neural Information Processing Systems*, 30, 2017.

[IW05] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 202–208, 2005.

[JKDG08] Prateek Jain, Brian Kulis, Inderjit S Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *NIPS*, volume 8, pages 761–768. Citeseer, 2008.

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[KNPW11] Daniel M Kane, Jelani Nelson, Ely Porat, and David P Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 745–754, 2011.

[KNW10] Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1161–1178. SIAM, 2010.

[LDFU13] Yichao Lu, Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems (NIPS)*, pages 369–377, 2013.

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L Nguyên, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–70. IEEE, 2016.

[LNRW19] Jerry Li, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. On mean estimation for general norms with statistical queries. In *Conference on Learning Theory (COLT)*, pages 2158–2172. PMLR, 2019.

[LSV18] Yin Tat Lee, Zhao Song, and Santosh S Vempala. Algorithmic theory of odes and sampling from well-conditioned logconcave densities. *arXiv preprint arXiv:1812.06243*, 2018.

[LW17] Kasper Green Larsen and Ryan Williams. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2182–2189. SIAM, 2017.

[MMR22] Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of johnson–lindenstrauss transform for k-means and k-medians clustering. *SIAM Journal on Computing*, (0):STOC19–269, 2022.

[MPS14] Andrew M McDonald, Massimiliano Pontil, and Dimitris Stamos. Spectral k-support norm regularization. *Advances in neural information processing systems*, 27, 2014.

[NS19] Vasileios Nakos and Zhao Song. Stronger l2/l2 compressed sensing; without iterating. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 289–297, 2019.

[Pag13] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013.

[Pel00] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3):167–176, 2000.

[Pri11] Eric Price. Efficient sketches for the set query problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 41–56, 2011.

[SS02] Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 360–369, 2002.

[SSSSC11] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[SSWZ22] Zhao Song, Baocheng Sun, Omri Weinstein, and Ruizhe Zhang. Sparse fourier transform over lattices: A unified approach to signal reconstruction. http://arxiv.org/abs/2205.00658, 2022.

[SSX21] Anshumali Shrivastava, Zhao Song, and Zhaozhuo Xu. Sublinear least-squares value iteration via locality sensitive hashing. *arXiv preprint arXiv:2105.08285*, 2021.

[SWY+19] Zhao Song, Ruosong Wang, Lin Yang, Hongyang Zhang, and Peilin Zhong. Efficient symmetric norm regression via linear sketching. *Advances in Neural Information Processing Systems*, 32, 2019.

[SWZ19] Zhao Song, David Woodruff, and Peilin Zhong. Towards a zero-one law for column subset selection. *Advances in Neural Information Processing Systems*, 32, 2019.

[SXZ22] Zhao Song, Zhaozhuo Xu, and Lichen Zhang. Speeding up sparsification with inner product search data structures. *arXiv preprint arXiv:2204.03209*, 2022.

[TZ12] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM J. Comput.*, 41(2):293–331, 2012.

[Val12] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 11–20. IEEE, 2012.

[WP11] Oren Weimann and David Peleg. A note on exact distance labeling. *Information processing letters*, 111(14):671–673, 2011.

[XSS21] Zhaozhuo Xu, Zhao Song, and Anshumali Shrivastava. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip data-structures. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] , we discuss the theoretical guarantee of the accuracy-efficiency tradeoff of our algorithm in Section 4 and Section E.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] . Our theoretical work does not have explicitly negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] , we explicitly stated the assumptions.
   (b) Did you include complete proofs of all theoretical results? [Yes] , please refer to Section 4, Section 5 and supplementary materials.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Roadmap. We divide the appendix into the following sections. Section A gives the preliminaries for our work. Section B introduces some useful properties of symmetric norms. Section C gives the proofs for the layer approximation technique. Section D states the formal version of our main theorem and algorithms. Section E gives more details about the time complexity proofs. Section F gives some details about the correctness proofs. Section G shows the space complexity of our algorithm. Section H states a streaming lower bound for the norm estimation problem and shows that our result is tight in this case. Section I gives an instantiation of the sparse recovery data structure (BATCHHEAVYHITTER) that simplifies the analysis in prior work.

A Preliminaries

In Section A.1, we define the notations we use in this paper. In Section A.2, we introduce some probability tools. In Section A.3, we define p-stable distributions.

A.1 Notations

For any positive integer n, we use $[n]$ to denote a set $\{1, 2, \cdots, n\}$. We use $\mathbb{E}[]$ to denote the expectation. We use $\Pr[]$ to denote the probability. We use $\mathrm{Var}[]$ to denote the variance. We define the unit vector $\xi^{(d_1)} := \frac{1}{\sqrt{d_1}}(1, 1, 1, , \ldots, 1, 0, \ldots, 0) \in \mathbb{R}^d$, for any $d_1 \in [d]$, which has d_1 nonzero coordinates. We abuse the notation to write $\xi^{(d_1)} \in \mathbb{R}^{d_1}$ by removing zero coordinates, and vice-versa by appending zeros. We use $\|x\|_2$ to denote entry-wise ℓ_2 norm of a vector. We use $\|x\|_{\mathrm{sym}}$ to denote the symmetric norm of a vector x.

We define tail as follows

Definition A.1 (Tail of a vector)**.** *For a given $x \in \mathbb{R}^d$ and an integer k, we use $x_{\overline{[k]}}$ or $x_{\mathrm{tail}(k)}$ to denote the vector that without largest top-k values (in absolute).*

A.2 Probability Tools

We state some useful inequalities in probability theory in below.

Lemma A.2 (Chernoff bound [Che52])**.** *Let $X = \sum_{i=1}^n X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^n p_i$. Then*

- $\Pr[X \geq (1+\delta)\mu] \leq \exp(-\delta^2\mu/3)$, $\forall \delta > 0$;
- $\Pr[X \leq (1-\delta)\mu] \leq \exp(-\delta^2\mu/2)$, $\forall 0 < \delta < 1$.

Lemma A.3 (Chebyshev's inequality)**.** *Let X be a random variable with finite expected value μ and finite non-zero variance σ^2. Then for any real number $k > 0$,*

$$\Pr[|X - \mu| \geq k\sigma] \leq \frac{1}{k^2}.$$

Theorem A.4 (Levy's isoperimetric inequality, [GMS86])**.** *For a continuous function $f : S^{d-1} \to \mathbb{R}$, Let M_f be the median of f, i.e., $\mu(\{x : f(x) \leq M_f\}) \geq 1/2$ and $\mu(\{x : f(x) \geq M_f\}) \geq 1/2$, where $\mu(\cdot)$ is the Haar probability measure on the unit sphere S^{d-1}. Then*

$$\mu(\{x : f(x) = M_f\}_\epsilon) \geq 1 - \sqrt{\pi/2}e^{-\epsilon^2 d/2},$$

where for a set $A \subset S^{d-1}$ we denote $A_\epsilon := \{x : l_2(x, A) \leq \epsilon\}$ and $l_2(x, A) := \inf_{y \in A} \|x - y\|_2$.

A.3 Stable Distributions

We define p-stable distributions.

Definition A.5 ([Ind06])**.** *A distribution $\mathcal{D}_p$ is called p-stable, if there exists $p \geq 0$ such that for any d real numbers $a_1, a_2, \ldots, a_d$ and i.i.d. variables $x_1, x_2, \ldots, x_d$ from distribution $\mathcal{D}_p$. the random variable $\sum_{i=1}^d a_i x_i$ has the same distribution as the variable $\|a\|_p y$, where y is a random variable from distribution $\mathcal{D}_p$.*

B Symmetric Norms

In this section, we give several technical tools for the symmetric norms. In Section B.1, we show the monotonicity of symmetric norms. In Section B.2, we show the concentration properties of symmetric norms. In Section B.3, we show some properties of the median of symmetric norm.

B.1 Monotonicity Property of Symmetric Norm

Lemma B.1 (Monotonicity of Symmetric Norms, see e.g. Proposition IV.1.1 in [Bha97]). *If $\|\cdot\|_{\mathrm{sym}}$ is a symmetric norm and $x, y \in \mathbb{R}^d$ satisfy that for all $i \in [d]$, $|x_i| \leq |y_i|$, then $\|x\|_{\mathrm{sym}} \leq \|y\|_{\mathrm{sym}}$.*

B.2 Concentration Property of Symmetric Norms

Let us give the results of concentration of measure as followed. The following tools and proofs can be found in [BBC+17]. However, for the completeness, we state them in below.

Lemma B.2 (Concentration of M_l)**.** *For every norm l on $\mathbb{R}^d$, if $x \in S^{d-1}$ is drawn uniformly at random according to Haar measure on the sphere, then*

$$\Pr[|\|x\|_l - M_l| > \frac{2\mathfrak{b}_l}{\sqrt{d}}] < \frac{1}{3}$$

Proof. With Theorem A.4, we know that, for a random x distributed according to the Haar measure on the l_2-sphere, with probability at least $1 - \sqrt{\pi/2}e^{-2} > \frac{2}{3}$, we can always find some $y \in S^{d-1}$, such that

$$\begin{aligned} \|x - y\|_2 &\leq \frac{2}{\sqrt{d}}, \text{ and} \\ \|y\|_l &= M_l. \end{aligned}$$

If we view the norm l as a function, then it is obvious that it is $\mathfrak{b}_l$-Lipschitz with respect to $\|\cdot\|_2$, so that we have

$$\begin{aligned} |\|x\|_l - M_l| &= |\|x\|_l - \|y\|_l| \\ &\leq \|x - y\|_l \\ &\leq \mathfrak{b}_l \|x - y\| \\ &\leq \frac{2\mathfrak{b}_l}{\sqrt{d}} \end{aligned}$$

where the first step follows from the definition of y, the second step follows from triangle inequality, the third step follows from that norm l is $\mathfrak{b}_l$-Lipschitz with respect to $\|\cdot\|_2$, and the last step follows from the definition of y.

Thus, we complete the proof. □

Lemma B.3 (Concentration inequalities for norms)**.** *For every $d > 0$, and norm l on $\mathbb{R}^n$, there is a vector $x \in S^{d-1}$ satisfying*

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

Proof. Let x be drawn uniformly randomly from a unit sphere. From Lemma B.2, we have

$$\begin{aligned} & \Pr[|\|x\|_l - M_{l_\infty^{(d)}}| > \frac{2}{\sqrt{d}}] < \frac{1}{3} \\ \textit{and} \quad & \Pr[|\|x\|_l - M_{l^{(d)}}| > \frac{2\mathfrak{b}_l}{\sqrt{d}}] < \frac{1}{3}. \end{aligned}$$

Define $\tau(x,t) := |\{i \in [d] : |x_i| < t\}|$. Now we are giving the proof to show that, for a constant K, over a choice of x, we have $\tau(x, \frac{1}{K\sqrt{d}}) < \frac{d}{2}$ with probability lager than $2/3$.

Let's consider an random vector $z \in \mathbb{R}^d$, such that each entry z_i is independent standard normal random variable. It is well known that $\frac{z}{\|z\|_2}$ is distributed uniformly over a sphere, so it has the same distribution as x. There is a universal constant K_1 such that

$$\Pr[\|z\|_2 > K_1\sqrt{d}] < \frac{1}{6},$$

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

$$\Pr[\tau(z, \frac{1}{K_2}) > \frac{d}{2}] < \frac{1}{6}.$$

By union bound, with probability larger than $2/3$, it holds simultaneously that

$$\|z\|_2 \ \leq K_1\sqrt{d} \quad \text{and} \quad \tau(z, \frac{1}{K_2}) \ < \frac{d}{2},$$

which imply:

$$\tau(\frac{z}{\|z\|_2}, \frac{1}{K_1K_2\sqrt{d}}) < \frac{d}{2}.$$

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

Thus, we complete the proof. □

B.3 Median of Symmetric Norm

We state a well-known fact before introducing Lemma B.5.

Fact B.4 (Concentration for median on infinity norm, [GMS86])**.** *There are absolute constants* $0 < \gamma_1 < \gamma_2$ *such that for every integer* $d \geq 1$*,*

$$\gamma_1\sqrt{\log(d)/d} \leq M_{l_\infty^{(d)}} \leq \gamma_2\sqrt{\log(d)/d}$$

We now give the following Lemma, which says that the l-norm of the (normalized) unit vector $\xi^{(d)}$ is closely related to the median of the norm. We are considering this vector because that it is related to a single layer of $\mathcal{L}$.

Lemma B.5 (Flat Median Lemma)**.** *Let* $l : \mathbb{R}^d \to \mathbb{R}$ *be a symmetric norm. Then*

$$\lambda_1 M_l/\sqrt{\log(d)} \leq l(\xi^{(d)}) \leq \lambda_2 M_l$$

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

We notice that the first inequality is tight for l_∞.

Proof. Using Lemma B.3, we can find a vector $x \in S^{d-1}$ and a constant $\lambda > 0$ satisfying

- $|\|x\|_\infty - M_{l_\infty}| \leq \lambda\sqrt{1/d}$
- $|\|x\|_l - M_l| \leq \lambda\mathfrak{b}_l/\sqrt{d}$, and
- $|\{i : |x_i| > \frac{1}{K\sqrt{d}}\}| > \frac{d}{2}$ for some universal constant K.

With Fact B.4, $M_{l_\infty} = \Theta(\sqrt{\log(d)/d})$. On the other hand, let $\mathrm{mmc}(l)$ be defined as Definition 1.3, we have

$$\mathrm{mmc}(l) \leq \gamma\sqrt{d}$$

for sufficiently small γ, thus

$$\frac{\lambda \mathfrak{b}_l}{\sqrt{d}} < M_l.$$

So with constants $\gamma_1, \gamma_2 > 0$, we have

$$\begin{aligned} \gamma_1 M_l &\leq \|x\|_l \leq \gamma_2 M_l \\ \textit{and} \quad \gamma_1\sqrt{\log(d)/d} &\leq \|x\|_\infty \leq \gamma_2\sqrt{\log(d)/d}. \end{aligned}$$

So that we have

$$|x| \leq \gamma_2\sqrt{\log(d)} \cdot \xi^{(d)},$$

where the inequality is entry-wise, and with monotonicity of symmetric norms (Lemma B.1), we have

$$\gamma_1 \leq \|x\|_l \leq \gamma_2\sqrt{\log(d)} \cdot \|\xi^{(d)}\|_l.$$

Now we move to the second condition of the lemma, we first let $M = \{i : |x_i| > \frac{1}{K\sqrt{d}}\}$. As $|M| > \frac{d}{2}$, we can find a permutation π satisfying

$$[d] - M \in \pi(M).$$

We define a vector $\pi(x)$ to be the vector by applying the permutation π to each entry of x. Denote by $|x|$ the vector of taking absolute value of x entry-wise. Notice $|x| + \pi(|x|) > \frac{\xi^{(d)}}{K}$ entry-wise, so that we have

$$\begin{aligned} \frac{1}{K} \cdot \|\xi^{(d)}\|_l &\leq \||x| + \pi(|x|)\|_l \\ &\leq \||x|\|_l + \|\pi(|x|)\|_l \\ &= 2\|x\|_l \\ &\leq 2\gamma_2 M_l, \end{aligned}$$

where the first step follows from the monotonicity of symmetric norm (Lemma B.1), the second step follows from the triangle inequality, the third step follows from the definition of π, and the last step follows from the definition of γ_2.

Thus, we complete the proof. □

The following lemma shows the monotonicity of the median (in d), a very useful property in the norm approximation.

Lemma B.6 (Monotonicity of Median)**.** *Let $l : \mathbb{R}^d \to \mathbb{R}$ be a symmetric norm. For all $d_1 \leq d_2$, where $d_1, d_2 \in [n]$, let $\mathrm{mmc}(l)$ be defined as Definition 1.3, we have*

$$M_{l^{(d_1)}} \leq \lambda \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d_1)} M_{l^{(d_2)}},$$

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

Proof. By Lemma B.5 and the fact that $\xi^{(d_1)}$ is also a vector in S^{d_2-1},

$$\lambda M_{l^{(d_1)}}/\sqrt{\log(d_1)} \leq \|\xi^{(d_1)}\|_l \leq \mathfrak{b}_{l^{(d_2)}} \leq \mathrm{mmc}(l) M_{l^{(d_2)}}.$$

Thus, we complete the proof. □

C Analysis of Layer Approximation

In this section, we show how to estimate the symmetric norm $l(\cdot)$ of a vector using the layer vectors. Section C.1 gives the definitions of layer vectors and important layers. Section C.2 shows that we can approximate the exact value of the norm and the layer vector. Section C.3 defines the contributing layer and shows its concentration property. Section C.4 proves that contributing layers are also important.

Throughout this section, let $\epsilon \in (0,1)$ be the precision, $\alpha > 0$ and $\beta \in (0,1]$ be some parameters depending on d, ϵ and $\mathrm{mmc}(l)$, where $\mathrm{mmc}(l)$ is defined as Definition 1.3. Furthermore, we assume $\mathrm{mmc}(l) \leq \gamma\sqrt{d}$, for constant parameter $0 \leq \gamma \ll 1/2$ small enough.[2]

C.1 Layer Vectors and Important Layers

Definition C.1 (Important Layers)**.** *For $v \in \mathbb{R}^d$, define layer $i \in \mathbb{N}_+$ as*

$$B_i := \{j \in [d] : \alpha^{i-1} < |v_j| \leq \alpha^i\},$$

and denote its size by $b_i := |B_i|$. We denote the number of non-zero b_i's by t, the number of non-empty layers. And we say that layer-i is β-important if

- $b_i > \beta \cdot \sum_{j=i+1}^{t} b_j$
- $b_i \alpha^{2i} \geq \beta \cdot \sum_{j \in [i]} b_j \alpha^{2j}$

With out loss of generality, We restrict the entries of the vector v to be in $[-m, m]$, and that $m = \mathrm{poly}(d)$. Then we know that the number of non-zero b_i's is at most $P = O(\log_\alpha(d))$. In the view of ℓ_2-norm, for an arbitrary vector $v \in \mathbb{R}^d$, if we normalize it to a unit vector, then the absolute value of each non-zero entry is at least $1/\mathrm{poly}(d)$. In order to simplify our analysis and algorithm for approximating $\|v\|_{\mathrm{sym}}$, we introduce the notations we use as follows.

Definition C.2 (Layer Vectors and Buckets)**.** *For each $i \in [P]$, let $\alpha^i \cdot \mathbf{1}_{b_i} \in \mathbb{R}^{b_i}$ denote a vector that has length b_i and every entry is α^i. Define the layer vector for $v \in \mathbb{R}^d$ with integer coordinates to be*

$$\mathcal{L}(v) := (\alpha^1 \cdot \mathbf{1}_{b_1}, \alpha^2 \cdot \mathbf{1}_{b_2}, \cdots, \alpha^P \cdot \mathbf{1}_{b_P}, 0 \cdot \mathbf{1}_{d - \sum_{j \in [P]} b_j}) \in \mathbb{R}^d;$$

and define the i-th bucket of $\mathcal{L}(v)$ to be

$$\mathcal{L}_i(v) := (0 \cdot \mathbf{1}_{b_1 + b_2 + \cdots + b_{i-1}}, \alpha^i \cdot \mathbf{1}_{b_i}, 0 \cdot \mathbf{1}_{d - \sum_{j \in [i]} b_j}) \in \mathbb{R}^d;$$

We also define $\mathcal{J}(v)$ and $\mathcal{J}_i(v)$ as above by replacing $\{b_i\}$ with the approximated values $\{c_i\}$. Denote $\mathcal{L}(v) \backslash \mathcal{L}_i(v)$ as the vector with the i-th bucket of $\mathcal{L}(v)$ replaced by 0. We also denote $(\mathcal{L}(v) \backslash \mathcal{L}_i(v)) \cup \mathcal{J}_i(v)$ as the vector by replacing the i-th bucket of $\mathcal{L}(v)$ with $\mathcal{J}_i(v)$, i.e.,

$$(\mathcal{L}(v) \backslash \mathcal{L}_i(v)) \cup \mathcal{J}_i(v) := (\alpha^1 \cdot \mathbf{1}_{b_1}, \alpha^2 \cdot \mathbf{1}_{b_2}, \cdots, \alpha^i \cdot \mathbf{1}_{c_i}, \cdots, \alpha^P \cdot \mathbf{1}_{b_P}, 0 \cdot \mathbf{1}_{d - \sum_{j \in [P]} b_j + b_i - c_i}) \in \mathbb{R}^d;$$

C.2 Approximated Layers Provides a Good Norm Approximation

We now proof that, $\|v\|_{\mathrm{sym}}$ can be approximated by using layer vector V. We first choose a base to be $\alpha := 1 + O(\epsilon)$.

Lemma C.3 (Approximattion with Layer Vector)**.** *For all $v \in \mathbb{R}^d$, we have*

$$\|\mathcal{L}(v)\|_{\mathrm{sym}}/\alpha \leq \|v\|_{\mathrm{sym}} \leq \|\mathcal{L}(v)\|_{\mathrm{sym}}.$$

Proof. The lemma follows from the monotonicity of symmetric norms(Lemma B.1) directly. □

The next key lemma shows that $\|\mathcal{J}(v)\|_{\mathrm{sym}}$ is a good approximation to $\|\mathcal{L}(v)\|_{\mathrm{sym}}$.

Lemma C.4 (Bucket Approximation)**.** *For every layer $i \in [P]$,*

[2] We note that beyond this regime, the streaming lower bound in Theorem H.2 implies that a linear-sized memory (time) is required to approximate the norm.

- *if* $c_i \leq b_i$*, then* $\|(\mathcal{L}(v)\backslash\mathcal{L}_i(v)) \cup \mathcal{J}_i(v)\|_{\text{sym}} \leq \|\mathcal{L}(v)\|_{\text{sym}}$*;*
- *if* $c_i \geq (1-\epsilon)b_i$*, then* $\|(\mathcal{L}(v)\backslash\mathcal{L}_i(v)) \cup \mathcal{J}_i(v)\|_{\text{sym}} \geq (1-\epsilon)\|\mathcal{L}(v)\|_{\text{sym}}$.

Proof. With the monotonicity of norm (Lemma B.1), the upper bound is quite obvious. So we just focus on the lower bound. Let us take the vector

$$\mathcal{J}_i(v) := (0 \cdot \mathbf{1}_{c_1+c_2+\cdots+c_{i-1}}, \alpha^i \cdot \mathbf{1}_{c_i}, 0, \cdots, 0) \in \mathbb{R}^d;$$

Here we define $\mathcal{K}(v) := \mathcal{L}(v) - \mathcal{L}_i(v)$. Then notice that $\mathcal{K}(v) + \mathcal{J}_i(v)$ is a permutation of the vector $(\mathcal{L}(v)\backslash\mathcal{L}_i(v)) \cup \mathcal{J}_i(v)$. We will then show that, under assumptions of the lemma, we have

$$\|\mathcal{K}(v) + \mathcal{J}_i(v)\|_{\text{sym}} \geq (c_i/b_i)\|\mathcal{L}(v)\|_{\text{sym}}.$$

Assume a vector $v \in \mathbb{R}^d$ and a permutation $\pi \in \Sigma_d$, we define a vector $\pi(x)$ to be the vector by applying the permutation π to each entry of x. Using the property of the symmetric norm, we have that $\|v\|_{\text{sym}} = \|\pi(v)\|_{\text{sym}}$. Consider a set of permutations that are cyclic shifts over the non-zero coordinates of $\mathcal{L}_i$, and do not move any other coordinates. That is, there is exactly b_i permutations in S, and for every $\pi \in S$, we have $\pi(\mathcal{K}(v)) = \mathcal{K}(v)$. By the construction of S, we have,

$$\sum_{\pi \in S} \pi(\mathcal{J}_i(v)) = c_i\mathcal{L}_i(v)$$

and therefore $\sum_{\pi \in S} \pi(\mathcal{K}(v) + \mathcal{J}_i(v)) = c_i\mathcal{L}_i(v) + b_i\mathcal{K}(v)$. As the vectors $\mathcal{L}_i(v)$ and $\mathcal{K}(v)$ have disjoint support, by monotonicity of symmetric norm (Lemma B.1) with respect to each coordinates we can deduce $\|c_i\mathcal{L}_i(v) + b_i\mathcal{K}(v)\|_{\text{sym}} \geq \|c_i(\mathcal{L}_i(v) + \mathcal{K}(v))\|_{\text{sym}}$. By plugging those together,

$$\begin{aligned}
c_i\|\mathcal{L}_i(v) + \mathcal{K}(v)\|_{\text{sym}} &\leq \|c_i\mathcal{L}_i(v) + b_i\mathcal{K}(v)\|_{\text{sym}} \\
&= \|\sum_{\pi \in S} \pi(\mathcal{J}_i(v) + \mathcal{K}(v))\|_{\text{sym}} \\
&\leq \sum_{\pi \in S} \|\pi(\mathcal{J}_i(v) + \mathcal{K}(v))\|_{\text{sym}} \\
&= b_i\|\pi(\mathcal{J}_i(v) + \mathcal{K}(v))\|_{\text{sym}}
\end{aligned}$$

where the first step follows from $c_i\|\mathcal{L}_i(v) + \mathcal{K}(v)\|_{\text{sym}} = \|c_i(\mathcal{L}_i(v) + \mathcal{K}(v))\|_{\text{sym}}$ and the monotonicity of the norm l, the second step follows from $\sum_{\pi \in S} \pi(\mathcal{J}_i(v) + \mathcal{K}(v)) = c_i\mathcal{L}_i(v) + b_i\mathcal{K}(v)$, the third step follows from triangle inequality, and the last step follows from the property of symmetric norm and $|S| = b_i$.

Hence,

$$\|\mathcal{J}_i(v) + \mathcal{K}(v)\|_{\text{sym}} \geq \frac{c_i}{b_i}\|\mathcal{L}(v)\|_{\text{sym}} \geq (1-\epsilon)\|\mathcal{L}(v)\|_{\text{sym}},$$

Thus, we complete the proof. □

C.3 Contributing Layers

Definition C.5 (Contributing Layers)**.** *For* $i \in [P]$*, layer* i *is called* β*-contributing if*

$$\|\mathcal{L}_i(v)\|_{\text{sym}} \geq \beta\|\mathcal{L}(v)\|_{\text{sym}}.$$

Lemma C.6 (Concentration with contributing layers)**.** *Let* $\mathcal{L}^*(v)$ *be the vector obtained from* V *by removing all layers that are not* β*-contributing. Then*

$$(1 - O(\log_\alpha(d)) \cdot \beta) \cdot \|\mathcal{L}(v)\|_{\text{sym}} \leq \|\mathcal{L}^*(v)\|_{\text{sym}} \leq \|\mathcal{L}(v)\|_{\text{sym}}.$$

Proof. Let $i_1, \ldots, i_k \in [P]$ be the layers that are not β- contributing.

Then we apply the triangle inequality and have,

$$\begin{aligned}
\|\mathcal{L}(v)\|_{\text{sym}} &\geq \|\mathcal{L}(v)\|_{\text{sym}} - \|\mathcal{L}_{i_1}(v)\|_{\text{sym}} - \cdots - \|\mathcal{L}_{i_k}(v)\|_{\text{sym}} \\
&\geq (1 - k_\beta)\|\mathcal{L}(v)\|_{\text{sym}}
\end{aligned}$$

The proof follows by bounding k by $P = O(\log_\alpha(n))$, which is the total number of non-zero b_i's. □

C.4 Contributing Layers Are Important

In this section, we give two lemmas to show that every β-contributing layer (Definition C.5) is β'-important (Definition C.1), where β' is depending on $\mathrm{mmc}(l)$ (Definition 1.3). The first property of the important layer is proved in Lemma C.7, and the second property is proved in Lemma C.8.

Lemma C.7 (Importance of contributing layers (Part 1))**.** *For $i \in [P]$, if layer i is β-contributing, then for some absolute constant $\lambda > 0$, we have*

$$b_i \geq \frac{\lambda \beta^2}{\mathrm{mmc}(l)^2 \log^2(d)} \cdot \sum_{j=i+1}^{P} b_j,$$

where $\mathrm{mmc}(l)$ is defined as Definition 1.3.

Proof. We first fix a layer i which is β-contributing. Let $\mathcal{U}(v)$ be the vector $\mathcal{L}(v)$ after removing buckets $j = 0, \ldots, i$. By Lemma B.5, there is an absolute constant $\lambda_1 > 0$ such that

$$\begin{aligned} \|\mathcal{L}_i(v)\|_{\mathrm{sym}} &= \alpha^i \cdot \sqrt{b_i} \cdot l(\xi^{(b_i)}) \\ &\leq \lambda_1 \cdot \alpha^i \cdot \sqrt{b_i} \cdot M_{l^{(b_i)}}, \end{aligned}$$

and similarly

$$\|\mathcal{U}(v)\|_{\mathrm{sym}} \geq \frac{\lambda_2 \alpha^i}{\sqrt{\log(d)}} \cdot (\sum_{j=i+1}^{P} b_j)^{1/2} M_{l^{(\sum_{j=i+1}^{P} b_j)}}.$$

With these two inequalities, we can have the following deduction.

First, we have

$$\|\mathcal{L}_i(v)\|_{\mathrm{sym}} \geq \beta \cdot \|\mathcal{L}(v)\|_{\mathrm{sym}} \geq \beta \cdot \|\mathcal{U}(v)\|_{\mathrm{sym}}.$$

Second, we assume that $b_i < \sum_{j=i+1}^{P} b_j$, as otherwise we are done:

$$b_i \geq \sum_{j=i+1}^{P} b_j \geq \frac{\lambda \beta^2}{\mathrm{mmc}(l)^2 \log^2(d)} \cdot \sum_{j=i+1}^{P} b_j.$$

Then, by the monotonicity of the median (Lemma B.6), we have

$$M_{l^{(b_i)}} \leq \lambda_3 \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d)} \cdot M_{l^{(\sum_{j=i+1}^{P} b_j)}}$$

for some absolute constant $\lambda_3 > 0$.

Putting it all together, we get

$$\beta \cdot \frac{\lambda_2 \alpha^i}{\sqrt{\log(d)}} \cdot (\sum_{j=i+1}^{P} b_j)^{1/2} \leq \lambda_1 \cdot \alpha^i \sqrt{b_i} \cdot \lambda_3 \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d)}.$$

Therefore, we finish the proof. □

Lemma C.8 (Importance of contributing layers (Part 2))**.** *For a symmetric l, let $\mathrm{mmc}(l)$ be defined as Definition 1.3. If layer $i \in [P]$ is β-contributing, then there is an absolute constant $\lambda > 0$ such that*

$$b_i \alpha^{2i} \geq \frac{\lambda \beta^2}{\mathrm{mmc}(l)^2 \cdot \log_\alpha(n) \cdot \log^2(n)} \cdot \sum_{j \in [i]} b_j \alpha^{2j}.$$

Proof. We first fix a layer i which is β-contributing, and let $h := \arg\max_{j \leq i} \sqrt{b_j} \alpha^j$. We consider the two different cases as follows.

First, if $b_i \geq b_h$ then the lemma follows obviously by

$$
\begin{aligned}
& \sum_{j \in [i]} b_j \alpha^{2j} \\
\leq & \, t \cdot b_h \cdot \alpha^{2h} \\
\leq & \, O(\log_\alpha(d)) \cdot b_i \cdot \alpha^{2i}.
\end{aligned}
$$

The second case is when $b_i < b_h$. With Definition C.5 and Lemma B.5, we have

$$
\begin{aligned}
& \lambda_1 \cdot \alpha^i \cdot \sqrt{b_i} \cdot M_{l(b_i)} \\
\geq & \, \|\mathcal{L}_i\|_{\mathrm{sym}} \\
\geq & \, \beta \cdot \|\mathcal{L}\|_{\mathrm{sym}} \\
\geq & \, \lambda_2 \cdot \beta \cdot \alpha^h \cdot \sqrt{\frac{b_h}{\log(d)}} \cdot M_{l(b_h)},
\end{aligned}
$$

for some absolute constants $\lambda_1, \lambda_2 > 0$, where the first step follows from Lemma B.5, the second step follows from Definition C.5, and the last step follows from Lemma B.5.

following from monotonicity of the median (Lemma B.6), we can plugging in $M_{l(b_i)} \leq \lambda_3 \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d)} \cdot M_{l(b_h)}$, for some absolute constant $\lambda_3 > 0$, so that we have

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

Square the above inequality and we can see that $b_h \cdot \alpha^{2h} \geq \frac{1}{O(\log_\alpha(d))} \cdot \sum_{j \in [i]} b_j \alpha^{2j}$.

Thus we complete the proof. □

D Formal Main Result and Algorithms

In this section, we state the formal version of our main theorem and algorithms. Section D.1 presents our main result: a data structure for distance estimation with symmetric norm. Section D.2 introduces the sparse recovery tools for sketching.

D.1 Formal Version of Our Main Result

Algorithm 3 Data structure for symmetric norm estimation

```
data structure DistanceOnSymmetricNorm                                   ▷ Theorem D.1

private:
procedure LayerVectorApprox(α, b_1, b_2, ..., b_P, d)                    ▷ Lemma C.3
    For each i ∈ [P], let α^i · 1_{b_i} ∈ R^{b_i} denote a vector that has length b_i and every entry is α^i
    L ← (α^1 · 1_{b_1}, ··· , α^P · 1_{b_P}, 0, ..., 0) ∈ R^d            ▷ Generate the layer vector
    return ||L||_sym                                                     ▷ Return the norm of the estimated layer vector
end procedure
end data structure
```

Here we divide function Query presented in Theorem 1.4 into two versions. One takes only the query point $q \in \mathbb{R}^d$ as input to ask all the distances. Another takes a query point $q \in \mathbb{R}^d$ and a set $\mathsf{S} \subset [n]$ to ask for the distance with the specific set of points. The latter can be viewed as a more general version of the former. In the former parts, we have proved the correctness of the query (Lemma 5.1), and the running time of version for all points (Lemma 4.3). Now we state the both in the following theorem, and the running time analysis for the latter will be stated in Section E.

Algorithm 4 Data structure for symmetric norm estimation: members, formal version of Algorithm 1

data structure DISTANCEONSYMMETRICNORM ▷ Theorem D.1
members
  $d, n \in \mathbb{N}_+$ ▷ n is the number of points, d is dimension
  $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$ ▷ Set of points being queried
  $L \in \mathbb{N}_+$ ▷ number of layers we subsample
  $R \in \mathbb{N}_+$ ▷ number of substreams in one layer
  ϵ, δ
  β ▷ used to cut important layer
  $U \in \mathbb{N}_+$ ▷ number of parallel processing
  BATCHHEAVYHITTER $\{S_{r,l,u}\}_{r \in [R], l \in [L], u \in [U]}$
  $\{H_{r,l,u} \subset [d] \times \mathbb{R}\}_{r \in [R], l \in [L], u \in [U]}$ ▷ each set H has a size of $2/\beta$, and is used to store the output of BATCHHEAVYHITTER
  γ ▷ parameter used when cutting layer vector
  $\mathrm{bmap} \in \{0,1\}^{R \times L \times U \times d}$
  $\overline{x}_{r,l,u} \in \mathbb{R}^{n \times d}$, for each $r \in [R], l \in [L], u \in [U]$ ▷ Substreams
end members
end data structure

Algorithm 5 Data structure for symmetric norm estimation: init, formal version of Algorithm 1

data structure DISTANCEONSYMMETRICNORM ▷ Theorem D.1
public:
procedure INIT($\{x_1, \cdots, x_i\} \subset \mathbb{R}^d, n \in \mathbb{N}_+, d \in \mathbb{N}_+, \delta \in (0, 0.1), \epsilon \in (0, 0.1)$) ▷ Lemma 4.1
  $n \leftarrow n$, $d \leftarrow d$, $\delta \leftarrow \delta$, $\epsilon \leftarrow \epsilon$
  for $i = 1 \to n$ **do**
    $x_i \leftarrow x_i$
  end for
  $\epsilon_1 \leftarrow O(\frac{\epsilon^2}{\log d})$ ▷ We define this notation for purpose of analysis
  $L \leftarrow \log(d)$, $R \leftarrow \Theta(\epsilon_1^{-2} \log(n/\delta) \log^2 d)$, $U \leftarrow \lceil \log(nd^2/\delta) \rceil$
  $\beta \leftarrow O(\frac{\epsilon^5}{\mathrm{mmc}(l)^2 \log^5 d})$
  for $r \in [R], l \in [L], u \in [U]$ **do**
    $S_{r,l,u}$.INIT($\sqrt{\beta}, n + 2, d$) ▷ Theorem D.3
  end for
  for $r \in [R], u \in [U], j \in [d], l \in [L]$ **do**
    Draw $\xi \in [0, 1]$
    if $\xi \in [0, 2^{-l}]$ **then**
      $\mathrm{bmap}[r, l, u, j] \leftarrow 1$
    else
      $\mathrm{bmap}[r, l, u, j] \leftarrow 0$
    end if
  end for
  for $r \in [R], u \in [U], i \in [n], j \in [d], l \in [L]$ **do**
    if $\mathrm{bmap}[r, l, u, j] = 1$ **then**
      $S_{r,l,u}$.ENCODESINGLE($i, j, x_{i,j}, d$) ▷ Theorem D.3
      $[\overline{x}_{r,l,u}]_{i,j} \leftarrow x_{i,j}$ ▷ Create a copy of subvectors
    else
      $[\overline{x}_{r,l,u}]_{i,j} \leftarrow 0$
    end if
  end for
end procedure
end data structure

Theorem D.1 (Main result, formal version of Theorem 1.4)**.** *There is a data structure (Algorithm 4, 5, 7, 6) uses $O(\epsilon^{-9} n (d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$ spaces for the Online Approximate Adaptive Symmetric Norm Distance Estimation Problem (Definition 1.1) with the following procedures:*

Algorithm 6 Data structure for symmetric norm estimation: update

data structure DISTANCEONSYMMETRICNORM ▷ Theorem D.1
public:
procedure UPDATE($i \in [n], z \in \mathbb{R}^d$) ▷ Lemma 4.2
▷ You want to replace x_i by z
  for $r \in [R], u \in [U], j \in [d], l \in [L]$ **do**
    if $\mathrm{bmap}[r, l, u] = 1$ **then**
      $S_{r,l,u}$.ENCODESINGLE(i, j, z_j, d) ▷ Theorem D.3
      $[\overline{x}_{r,l,u}]_{i,j} \leftarrow z_j$ ▷ Create a copy of subvectors
    else
      $[\overline{x}_{r,l,u}]_{i,j} \leftarrow 0$
    end if
  end for
end procedure
end data structure

- INIT($\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d, \epsilon \in (0,1), \delta \in (0,1)$)*: Given n data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter ϵ and a failure probability δ as input, the data structure preprocesses in time $O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$.*

- UPDATEX($z \in \mathbb{R}^d, i \in [n]$)*: Given an update vector $z \in \mathbb{R}^d$ and index $i \in [n]$, the* UPDATEX *takes z and i as input and updates the data structure with the new i-th data point in $O(\epsilon^{-4} d \log^9(nd/\delta))$ time.*

- QUERY($q \in \mathbb{R}^d$) *(Querying all points): Given a query point $q \in \mathbb{R}^d$, the* QUERY *operation takes q as input and approximately estimates the symmetric norm distances from q to all the data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ in time*

$$O(\epsilon^{-9}(d + n \cdot \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$$

  i.e. it provides a set of estimates $\{\mathrm{dst}_i\}_{i=1}^n$ such that:

$$\forall i \in [n], (1-\epsilon)\|q - x_i\|_{\mathrm{sym}} \leq \mathrm{dst}_i \leq (1+\epsilon)\|q - x_i\|_{\mathrm{sym}}$$

  with probability at least $1 - \delta$.

- QUERY($q \in \mathbb{R}^d, \mathsf{S} \subseteq [n]$) *(Querying a specific set S of points). Given a query point $q \in \mathbb{R}^d$ and an index $i \in [n]$, the* QUERYONE *operation takes q and i as input and approximately estimates the symmetric norm distances from q to the i-th point $x_i \in \mathbb{R}^d$ in time*

$$O(\epsilon^{-9}(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$$

  i.e. it provides a estimated distance $\mathrm{dst} \in \mathbb{R}^{\mathsf{S}}$ such that:

$$(1-\epsilon)\|q - x_i\|_{\mathrm{sym}} \leq \mathrm{dst}_i \leq (1+\epsilon)\|q - x_i\|_{\mathrm{sym}}, \forall i \in \mathsf{S}$$

  with probability at least $1 - \delta$.

- ESTPAIR($i, j \in [n]$) *Given indices $i, j \in [n]$, the* ESTPAIR *operation takes i and j as input and approximately estimates the symmetric norm distances from i-th to the j-th point $x_i, x_j \in \mathbb{R}^d$ in time*

$$O(\epsilon^{-9} \cdot \mathrm{mmc}(l)^2 \log^{14}(nd/\delta))$$

  i.e. it provides a estimated distance pair *such that:*

$$(1-\epsilon)\|x_i - x_j\|_{\mathrm{sym}} \leq \mathrm{pair} \leq (1+\epsilon)\|x_i - x_j\|_{\mathrm{sym}}$$

  with probability at least $1 - \delta$.

Proof. In Lemma E.3, Lemma E.4, Lemma E.1 and Lemma E.2 we analyze the running time for INIT, UPDATE and QUERY and ESTPAIR respectively.

Algorithm 7 Data structure for symmetric norm estimation: query, formal version of Algorithm 2

data structure DISTANCEONSYMMETRICNORM ▷ Theorem D.1
procedure QUERY($q \in \mathbb{R}^d$) ▷ Lemma 4.3, 5.1
  $P \leftarrow O(\log_\alpha(d))$ ▷ P denotes the number of non-empty layer sets
  for $r \in [R], l \in [L], u \in [U]$ **do**
    $S_{r,l,u}$.ENCODE$(n+1, q, d)$ ▷ Generate Sketch for q
  end for
  $\xi \leftarrow$ chosen uniformly at random from $[1/2, 1]$
  $\gamma \leftarrow \Theta(\epsilon)$
  $\alpha \leftarrow 1 + \gamma \cdot \xi$
  $P \leftarrow O(\log_\alpha(d)) = O(\log(d)/\epsilon)$ ▷ P denotes the number of non-empty layer sets
  for $i \in [n]$ **do**
    for $r \in [R], l \in [L], u \in [U]$ **do**
      $S_{r,l,u}$.SUBTRACT$(n+2, n+1, i)$
      $H_{r,l,u} \leftarrow S_{r,l,u}$.DECODE$(n+2, \sqrt{\beta}, d)$ ▷ This can be done in $\frac{2}{\beta}\,\mathrm{poly}(\log d)$
      ▷ At this point $H_{r,l,u}$ is a list of index, the value of each index is reset to 0
      for $k \in H_{r,l,u}$ **do**
        value $\leftarrow [\overline{x}_{r,l,u}]_{i,k}$
        $H_{r,l,u}[k] \leftarrow$ value
        $w \leftarrow \lceil \log(\text{value})/\log(\alpha) \rceil$
        if $\alpha^{w-1} \geq \text{value}/(1+\epsilon)$ **then**
          $H_{r,l,u} \leftarrow$ null
          break
        end if
      end for
      if $H_{r,l,u} \neq$ null **then**
        $H_{r,l} \leftarrow H_{r,l,u}$
      end if
    end for
    for $l \in [L], k \in [P]$ **do**
      $A^i_{l,k} \leftarrow |\{k \mid \exists k \in H_{r,l}, \alpha^{k-1} < |H_{r,l}[k]| \leq \alpha^k\}|$
    end for
    for $k \in [P]$ **do**
      $q^i_k \leftarrow \max_{l \in [L]} \{l \mid A^i_{l,k} \geq \frac{R \log(1/\delta)}{100 \log(d)}\}$ ▷ Definition F.7
      If q^i_k does not exist, then $\widehat{\eta}^i_k \leftarrow 0$; Else $\widehat{\eta}^i_k \leftarrow \frac{A_{q^i_k,k}}{R(1+\epsilon_1)}$
      If $\widehat{\eta}^i_k = 0$ then $c^i_k \leftarrow 0$; Else $c^i_k \leftarrow \frac{\log(1-\widehat{\eta}^i_k)}{1-w^{-q_k}}$
    end for
    $\mathrm{dst}_i \leftarrow$ LAYERVETCORAPPROX$(\alpha, c^i_1, c^i_2, \ldots, c^i_P, d)$
    for $r \in [R], l \in [L], u \in [U]$ **do**
      $\{H_{r,l,u}\} \leftarrow \{0\}$ ▷ Reset the sets to use for next point
    end for
  end for
  return $\{\mathrm{dst}_i\}_{i \in [n]}$
end procedure
end data structure

Lemma G.1 shows the space complexity.

In Lemma 5.1 we give the correctness of QUERY, and correctness for ESTPAIR follows directly.

Thus, putting them all together, we prove the Theorem.

□

D.2 Sparse Recovery tools

We start with describing a data structure problem

Definition D.2 (Batch Heavy Hitter)**.** *Given an $n \times d$ matrix, the goal is to design a data structure that supports the following operations:*

- INIT($\epsilon \in (0, 0.1), n, d$). *Create a set of Random Hash functions and all the n copies of sketches share the same hash functions.*
- ENCODE($i \in [n], z \in \mathbb{R}^d, d$). *This step encodes z into i-th sketched location and store a size $\mathcal{S}_{\mathrm{space}}$ linear sketch.*
- ENCODESINGLE($i \in [n], j \in [d], z \in \mathbb{R}, d$). *This step updates one sparse vector $e_j z \in \mathbb{R}^d$ into i-th sketched location.*
- SUBTRACT($i, j, l \in [n]$). *This function updates the sketch at i-th location by j-th sketch minus l-th sketch.*
- DECODE($i \in [n], \epsilon \in (0, 0.1), d$). *This function returns a set $L \subseteq [d]$ of size $|L| = O(\epsilon^{-2})$ containing all ϵ-heavy hitters $i \in [n]$ under ℓ_p. Here we say i is an ϵ-heavy hitter under ℓ_2 if $|x_i| \geq \epsilon \cdot \|x_{\overline{[\epsilon^{-2}]}}\|_2$ where $x_{\overline{[k]}}$ denotes the vector x with the largest k entries (in absolute value) set to zero. Note that the number of heavy hitters never exceeds $2/\epsilon^2$.*

The existing work [KNPW11, Pag13] implies the following result. However their proofs are very decent and complicated. We provide another data structure in Section I that significantly simplifies the analysis (by only paying some extra log factors). We believe it is of independent interest.

Theorem D.3. *There is (linear sketch) data structure* BATCHHEAVYHITTER(ϵ, n, d) *that uses $\mathcal{S}_{\mathrm{space}}$ space that support the following operations:*

- INIT($\epsilon \in (0, 0.1), n, d$). *This step takes $\mathcal{T}_{\mathrm{init}}(\epsilon, n, d)$ time.*
- ENCODE($i \in [n], z \in \mathbb{R}^d, d$). *This step takes $\mathcal{T}_{\mathrm{encode}}(d)$ time.*
- ENCODESINGLE($i \in [n], j \in [d], z \in \mathbb{R}, d$). *This step takes $\mathcal{T}_{\mathrm{encodesingle}}(d)$ time.*
- SUBTRACT($i, j, l \in [n]$). *This step takes $\mathcal{T}_{\mathrm{encodesingle}}(d)$ time.*
- DECODE($i \in [n], \epsilon \in (0, 0.1), d$). *This step takes $\mathcal{T}_{\mathrm{decode}}(\epsilon, d)$ time.*

The running time of function can be summarize as

- $\mathcal{S}_{\mathrm{space}}(\epsilon, d) = n \cdot O(\epsilon^{-2} \log^2 d)$
- $\mathcal{T}_{\mathrm{init}}(\epsilon, n, d) = n \cdot O(\epsilon^{-2} \log^2 d)$
- $\mathcal{T}_{\mathrm{encode}}(d) = O(d \log^2(d))$
- $\mathcal{T}_{\mathrm{encodesingle}}(d) = O(\log^2(d))$
- $\mathcal{T}_{\mathrm{subtract}}(\epsilon, d) = O(\epsilon^{-2} \log^2 d)$
- $\mathcal{T}_{\mathrm{decode}}(\epsilon, d) = O(\epsilon^{-2} \log^2 d)$

Note that the succeed probability is at least 0.99.

We remark that, to boost the probability from constant to $1 - 1/\operatorname{poly}(nd)$ we just need to pay an extra $\log(nd)$ factor.

E More Details of the Time Complexity

In this section, we analyze the running time of the general version of QUERY. Lemma 4.3 is a special case of the following Lemma when $\mathsf{S} = [n]$.

Lemma E.1 (QUERY time for general version)**.** *Given a query point $q \in \mathbb{R}^d$ and a set $\mathsf{S} \subseteq [n]$, the procedure* QUERY *(Algorithm 8) runs in time*

$$O(\epsilon^{-9}(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \log^{14}(nd/\delta)).$$

Proof. The QUERY operation for a stored vector (Algorithm 8) has the following two parts:

Algorithm 8 Data structure for symmetric norm estimation: query set

1: **data structure** DISTANCEONSYMMETRICNORM ▷ Theorem D.1
2: **procedure** QUERY($q \in \mathbb{R}^d$, $\mathsf{S} \subseteq [n]$) ▷ Lemma 4.3, 5.1
3: $P \leftarrow O(\log_\alpha(d))$ ▷ P denotes the number of non-empty layer sets
4: **for** $r \in [R], l \in [L], u \in [U]$ **do**
5: $S_{r,l,u}$.ENCODE$(n+1, q, d)$ ▷ Generate Sketch for q
6: **end for**
7: $\xi \leftarrow$ chosen uniformly at random from $[1/2, 1]$
8: $\gamma \leftarrow \Theta(\epsilon)$
9: $\alpha \leftarrow 1 + \gamma \cdot \xi$
10: $P \leftarrow O(\log_\alpha(d)) = O(\log(d)/\epsilon)$ ▷ P denotes the number of non-empty layer sets
11: **for** $i \in \mathsf{S}$ **do**
12: **for** $r \in [R], l \in [L], u \in [U]$ **do**
13: $S_{r,l,u}$.SUBTRACT$(n+2, n+1, i)$
14: $H_{r,l,u} \leftarrow S_{r,l,u}$.DECODE$(n+2, \sqrt{\beta}, d)$ ▷ This can be done in $\frac{2}{\beta}\,\mathrm{poly}(\log d)$
15: ▷ At this point $H_{r,l,u}$ is a list of index, the value of each index is reset to 0
16: **for** $k \in H_{r,l,u}$ **do**
17: value $\leftarrow [\overline{x}_{r,l,u}]_{i,k}$
18: $H_{r,l,u}[k] \leftarrow$ value
19: $w \leftarrow \lceil \log(\text{value})/\log(\alpha) \rceil$
20: **if** $\alpha^{w-1} \geq \text{value}/(1+\epsilon)$ **then**
21: $H_{r,l,u} \leftarrow$ null
22: **break**
23: **end if**
24: **end for**
25: **if** $H_{r,l,u} \neq$ null **then**
26: $H_{r,l} \leftarrow H_{r,l,u}$
27: **end if**
28: **end for**
29: **for** $l \in [L], k \in [P]$ **do**
30: $A^i_{l,k} \leftarrow |\{k \mid \exists k \in H_{r,l}, \alpha^{k-1} < |H_{r,l}[k]| \leq \alpha^k\}|$
31: **end for**
32: **for** $k \in [P]$ **do**
33: $q^i_k \leftarrow \max_{l \in [L]} \{l \mid A^i_{l,k} \geq \frac{R\log(1/\delta)}{100\log(d)}\}$
34: If q^i_k does not exist, then $\widehat{\eta}^i_k \leftarrow 0$; Else $\widehat{\eta}^i_k \leftarrow \frac{A_{q^i_k,k}}{R(1+\epsilon_2)}$
35: If $\widehat{\eta}^i_k = 0$ then $c^i_k \leftarrow 0$; Else $c^i_k \leftarrow \frac{\log(1-\widehat{\eta}^i_k)}{1-w^{-q_k}}$
36: **end for**
37: $\mathrm{dst}_i \leftarrow$ LAYERVETCORAPPROX$(\alpha, c^i_1, c^i_2, \ldots, c^i_P, d)$
38: **for** $r \in [R], l \in [L], u \in [U]$ **do**
39: $\{H_{r,l,u}\} \leftarrow \{0\}$ ▷ Reset the sets to use for next point
40: **end for**
41: **end for**
42: **return** $\{\mathrm{dst}_i\}_{i \in \mathsf{S}}$
43: **end procedure**
44: **end data structure**

- **Part 1:** Line 5 takes $O(RLU \cdot \mathcal{T}_{\mathrm{encode}})$ time to call ENCODE to generate sketches for q.
- **Part 2:** For each $i \in \mathsf{S}$
  - Line 13 takes $O(RLU \cdot \mathcal{T}_{\mathrm{subtract}})$ time to compute sketch of the difference between q and x_i, and store the sketch at index of $n+2$.
  - Line 14 takes $O(RLU \cdot \mathcal{T}_{\mathrm{decode}})$ time to decode the BATCHHEAVYHITTER and get estimated heavy hitters of $q - x_i$.
  - Line 16 to Line 24 takes $O(RLU \cdot 2/\beta)$ time to analyze the BATCHHEAVYHITTER and get the set of indices, where $2/\beta$ is the size of the set.
  - Line 30 takes $O(LP \cdot 2/\beta)$ time to compute size of the layer sets cut by α.
  - Line 32 to Line 36 takes $O(PL)$ time to compute the estimation of each layer.

Algorithm 9 Data structure for symmetric norm estimation: query pair

data structure DISTANCEONSYMMETRICNORM ▷ Theorem D.1
procedure ESTPAIR($i \in [n], j \in [n]$) ▷ Lemma 4.3, 5.1
  $P \leftarrow O(\log_\alpha(d))$ ▷ P denotes the number of non-empty layer sets
  $\xi \leftarrow$ chosen uniformly at random from $[1/2, 1]$
  $\gamma \leftarrow \Theta(\epsilon)$
  $\alpha \leftarrow 1 + \gamma \cdot \xi$
  $P \leftarrow O(\log_\alpha(d)) = O(\log(d)/\epsilon)$ ▷ P denotes the number of non-empty layer sets
  for $r \in [R], l \in [L], u \in [U]$ **do**
    $S_{r,l,u}$.SUBTRACT$(n+2, j, i)$
    $H_{r,l,u} \leftarrow S_{r,l,u}$.DECODE$(n+2, \sqrt{\beta}, d)$ ▷ This can be done in $\frac{2}{\beta}\,\mathrm{poly}(\log d)$
    ▷ At this point $H_{r,l,u}$ is a list of index, the value of each index is reset to 0
    for $k \in H_{r,l,u}$ **do**
      value $\leftarrow [\overline{x}_{r,l,u}]_{i,k}$
      $H_{r,l,u}[k] \leftarrow$ value
      $w \leftarrow \lceil \log(\text{value}) / \log(\alpha) \rceil$
      if $\alpha^{w-1} \geq \text{value}/(1+\epsilon)$ **then**
        $H_{r,l,u} \leftarrow$ null
        break
      end if
    end for
    if $H_{r,l,u} \neq$ null **then**
      $H_{r,l} \leftarrow H_{r,l,u}$
    end if
  end for
  for $l \in [L], k \in [P]$ **do**
    $A_{l,k} \leftarrow |\{k \mid \exists k \in H_{r,l}, \alpha^{k-1} < |H_{r,l}[k]| \leq \alpha^k\}|$
  end for
  for $k \in [P]$ **do**
    $q_k \leftarrow \max_{l \in [L]} \{l \mid A_{l,k} \geq \frac{R \log(1/\delta)}{100 \log(d)}\}$
    If q_k does not exist, then $\widehat{\eta}_k \leftarrow 0$; Else $\widehat{\eta}_k \leftarrow \frac{A_{q_k,k}}{R(1+\epsilon_2)}$
    If $\widehat{\eta}_k = 0$ then $c_k \leftarrow 0$; Else $c_k \leftarrow \frac{\log(1-\widehat{\eta}_k)}{1-w^{-q_k}}$
  end for
  dst $\leftarrow$ LAYERVETCORAPPROX$(\alpha, c_1, c_2, \ldots, c_P, d)$
  return dst
end procedure
end data structure

The total running time of this part is:

$$
\begin{aligned}
& |\mathsf{S}| \cdot (O(RLU \cdot \mathcal{T}_{\mathrm{subtract}}) + O(RLU \cdot \mathcal{T}_{\mathrm{decode}}) + O(RLU \cdot 2/\beta) + O(LP \cdot 2/\beta) + O(LP))) \\
= & O(|\mathsf{S}| \cdot L(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1}))
\end{aligned}
$$

time in total.

Taking these two parts together we have the total running time of the QUERY procedure:

$$
\begin{aligned}
& O(RLU \cdot \mathcal{T}_{\mathrm{encode}}) + O(|\mathsf{S}| \cdot L(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1})) \\
= & O(RLU(\mathcal{T}_{\mathrm{encode}} + |\mathsf{S}| \cdot (\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1})) + |\mathsf{S}| \cdot LP\beta^{-1}) \\
= & O(\epsilon^{-4} \log^6(nd/\delta)(d \log^2(d) \log(nd) + |\mathsf{S}| \cdot \beta^{-1} \log^2(d) \log(nd)) + |\mathsf{S}| \cdot \epsilon^{-1} \log^2(d) \beta^{-1}) \\
= & O(\epsilon^{-9}(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))
\end{aligned}
$$

where the first step follows from the property of big O notation, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{subtract}}, \mathcal{T}_{\mathrm{decode}}$ (Theorem D.3) , P, the third step follows from merging the terms.

Thus, we complete the proof.

□

Lemma E.2 (EstPair time). *Given a query point $q \in \mathbb{R}^d$, the procedure* EstPair *(Algorithm 9) runs in time*

$$O(\epsilon^{-9} \cdot \mathrm{mmc}(l)^2 \log^{14}(nd/\delta)).$$

Proof. The EstPair operation (Algorithm 9) has the following two parts:

- Line 9 takes $O(RLU \cdot \mathcal{T}_{\mathrm{subtract}})$ time to compute sketch of the difference between x_i and x_j, and store the sketch at index of $n+2$.
- Line 10 takes $O(RLU \cdot \mathcal{T}_{\mathrm{decode}})$ time to decode the BatchHeavyHitter and get estimated heavy hitters of $x_i - x_j$.
- Line 12 to Line 20 takes $O(RLU \cdot 2/\beta)$ time to analyze the BatchHeavyHitter and get the set of indices, where $2/\beta$ is the size of the set.
- Line 26 takes $O(LP \cdot 2/\beta)$ time to compute size of the layer sets cut by α.
- Line 28 to Line 32 takes $O(PL)$ time to compute the estimation of each layer.

The total running time is:

$$\begin{aligned}
& O(RLU \cdot \mathcal{T}_{\mathrm{subtract}}) + O(RLU \cdot \mathcal{T}_{\mathrm{decode}}) + O(RLU \cdot 2/\beta) + O(LP \cdot 2/\beta) + O(LP) \\
= & O(RLU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + LP\beta^{-1}) \\
= & O(\epsilon^{-4} \log^6(nd/\delta)(\beta^{-1} \log^2(d) \log(nd) + \epsilon^{-1} \log^2(d)\beta^{-1} \log(nd)) \\
= & O(\epsilon^{-9} \cdot \mathrm{mmc}(l)^2 \log^{14}(nd/\delta))
\end{aligned}$$

time in total, where the first step follows from the property of big O notation, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{subtract}}, \mathcal{T}_{\mathrm{decode}}$ (Theorem D.3) , P, the third step follows from merging the terms.

Thus, we complete the proof.

□

Lemma E.3 (Init time, formal version of Lemma 4.1). *Given data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter $\epsilon > 0$, and a failure probability $\delta > 0$ as input, the procedure* Init *(Algorithm 5) runs in time*

$$O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta)).$$

Proof. The Init time includes these parts:

- Line 12 takes $O(RLU \cdot \mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d))$ to initialize sketches
- Line 17 to Line 19 takes $O(RUdL)$ to generate the bmap;
- Line 24 takes $O(ndRUL \cdot \mathcal{T}_{\mathrm{encodesingle}}(d))$ to generate sketches

By Theorems D.3, we have

- $\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d)) = n \cdot O(\beta^{-1} \log^2(d) \log(nd)) = O(n \cdot \mathrm{mmc}(l)^2 \log^7(d) \log(nd)\epsilon^{-5})$,
- $\mathcal{T}_{\mathrm{encodesingle}}(d) = O(\log^2(d) \log(nd))$.

Adding them together we got the time of

$$\begin{aligned}
& O(RLU\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d)) + O(RUdL) + O(ndRUL \cdot \mathcal{T}_{\mathrm{encodesingle}}(d)) \\
= & O(RLU(\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d) + nd \cdot \mathcal{T}_{\mathrm{encodesingle}}(d))) \\
= & O(\epsilon^{-4} \log(d/\delta) \log^4(d) \cdot \log(d) \cdot \log(nd^2/\delta) \log(nd)(n \cdot \mathrm{mmc}(l)^2 \log^7(d)\epsilon^{-5} + nd \log^2(d)))
\end{aligned}$$

$= O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta)),$

where the first step follows from merging the terms, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encodesingle}}(d), \mathcal{T}_{\mathrm{init}}$, the third step follows from merging the terms.

Thus, we complete the proof. □

Lemma E.4 (UPDATE time, formal version of Lemma 4.2)**.** *Given a new data point $z \in \mathbb{R}^d$, and an index i where it should replace the original data point $x_i \in \mathbb{R}^d$. The procedure* UPDATE *(Algorithm 6) runs in time*

$$O(\epsilon^{-4} d \log^9(nd/\delta)).$$

Proof. The UPDATE operation calls BATCHHEAVYHITTER.ENCODE for RLU times, so it has the time of

$$\begin{aligned} O(RLU \cdot \mathcal{T}_{\mathrm{encode}}(d)) &= O(\epsilon^{-4} \log(n/\delta) \log^4(d) \cdot \log(d) \cdot \log(nd^2/\delta) \cdot d \log^2(d) \log(nd)) \\ &= O(\epsilon^{-4} d \log^9(nd/\delta)) \end{aligned}$$

where the first step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}(d)$, the second step follows from

$$\begin{aligned} & \log(n/\delta) \log^4(d) \log(d) \log(nd^2/\delta) \log^2(d) \log(nd) \\ = & (\log(n/\delta))(\log^7 d)(2 \log d + \log(n/\delta)) \log(nd) \\ = & O(\log^9(nd/\delta)) \end{aligned}$$

Thus, we complete the proof. □

F More Details of the Correctness Proofs

In this section, we give the complete proofs of the correctness of our algorithms. In Section F.1 we state and prove the main result of this section, using the technical lemmas in the following subsections. In Section F.2, we define the trackable layers and show the connection with important layers (Definition C.1). In Section F.3, we analyze the sample probability and track probability of a layer. In Section F.4, we show that a good estimation of track probability implies a good approximation of the layer size, which completes the proof of the correctness.

F.1 Correctness of Layer Size Estimation

We first show that, the estimation of layer sizes output by our data structure are good to approximate the exact values.

Lemma F.1 (Correctness of layer size approximation)**.** *We first show that, the layer sizes $c_1^i, c_2^i, \ldots, c_P^i$ our data structure return satisfy*

- *for all $k \in P$, $c_k^i \leq b_k^i$;*
- *if k is a β-important layer (Definition C.1) of $q - x_i$, then $c_k^i \geq (1 - \epsilon_1) b_k^i$,*

with probability at least $1 - \delta$, where the b_k^i is the ground truth k-th layer size of $q - x_i$.

Proof. We first define two events as

- E_1: for all important layers $k \in [P]$, q_k is well defined;
- E_2: for all $k \in [P]$, if $\widehat{\eta}_k > 0$ then $(1 - O(\epsilon)) \eta'_{k,q_k} \leq \widehat{\eta}_k \leq \eta'_{k,q_k}$.

With Lemma F.6 and Lemma F.8, we have that

$$\Pr[E_1 \cap E_2] \geq 1 - \delta^{O(\log(d))}.$$

When the output of every BATCHHEAVYHITTER is correct, if follows from Lemma F.5, Lemma F.6 and Lemma F.9 the algorithm outputs an approximation to the layer vector meeting the two criteria.

The BATCHHEAVYHITTER is used a total of LR times, each with error probability at most δ/d. By a union bound over the layers, the failure probability is at most $(\mathrm{poly}(\log d)) \cdot \delta/d = o(\delta)$. Therefore, the total failure probability of the algorithm is at most $1 - o(\delta)$.

Thus, we complete the proof. □

F.2 Trackability of Layers

Definition F.2 (Trackability of layers)**.** *A layer $k \in [P]$ of a vector x is β-trackable, if*

$$\alpha^{2k} \geq \beta \cdot \|x_{\overline{[\beta^{-1}]}}\|_2^2$$

where $x_{\overline{[\kappa]}}$ is defined as Definition A.1.

Lemma F.3 (Importance and Trackability)**.** *Let α be some parameter such that $\alpha \in [0, 2]$. Suppose subvector $\widetilde{x}$ is obtained by subsampling the original vector with probability p.*

If $k \in [P]$ is a β-important layer, then for any $\lambda > P$, with probability at least $1 - P\exp(-\frac{\lambda p b_k}{P\beta})$, layer k is $\frac{\beta}{\lambda p b_k}$-trackable.

In particular, if $pb_k = O(1)$, then with probability at least $1 - P\exp(-\Omega(\frac{\lambda}{P\beta}))$, layer k is $\frac{\beta}{\lambda}$-trackable.

Proof. Let $(\zeta_0, \zeta_1, \dots)$ be the new level sizes of the subvector $\widetilde{x}$ we sampled. Thus, for $k \in [P]$, we have

$$\mathbb{E}[\zeta_k] = p \cdot b_k.$$

By the definition of important layer (Definition C.1) we have

$$\mathbb{E}[\zeta_k] \geq \beta \cdot \mathbb{E}\Big[\sum_{j=k+1}^{P} \zeta_j\Big]$$

and

$$\mathbb{E}[\zeta_k \cdot \alpha^{2k}] \geq \beta \cdot \mathbb{E}\Big[\sum_{j\in[k]} \zeta_j \cdot \alpha^{2j}\Big].$$

To have layer k trackable, it has to be in the top $\lambda p b_k/\beta$ elements, so that we have $\sum_{j=k+1}^{P} \zeta_j \leq \lambda p b_k/\beta$. By Chernoff bound (Lemma A.2), we can know the probability that this event does not happen is

$$\Pr\Big[\sum_{j=k+1}^{P} b_j > \frac{\lambda p b_k}{\beta}\Big] \leq \exp(-\Omega(\lambda p b_k/\beta)).$$

On the other hand, for level k to be trackable $\alpha^{2k} \geq \frac{\beta}{\lambda p b_k}\sum_{j\in[k]} \zeta_j \alpha^{2j}$.

Thus, the complement occurs with probability

$$\begin{aligned}\Pr\Big[\frac{\beta}{\lambda p b_k}\sum_{j\in[k]} \zeta_j\alpha^{2j} > \alpha^{2k}\Big] &\leq \Pr\Big[\exists j, \zeta_j\alpha^{2j} \geq \frac{\lambda p b_k \alpha^{2k}}{P\beta}\Big] \\ &\leq \sum_{j\in[P]} \Pr\Big[\zeta_j\alpha^{2j} \geq \frac{\lambda p b_k \alpha^{2k}}{P\beta}\Big].\end{aligned}$$

By Chernoff bound (Lemma A.2) and the fact that $\mathbb{E}[\zeta_j\alpha^{2j}] \leq p b_k \alpha^{2k}$, we have

$$\begin{aligned}\Pr\Big[\frac{\beta}{\lambda p b_k}\sum_{j\in[k]} \zeta_j\alpha^{2j} > \alpha^{2k}\Big] &\leq P \cdot \exp\Big(-\Omega(\frac{\lambda p b_k \alpha^{2k}}{\alpha^{2j} P\beta})\Big) \\ &\leq P \cdot \exp\Big(-\Omega(\frac{\lambda p b_k}{\beta})\Big).\end{aligned}$$

Thus we complete the proof. □

F.3 Probability Analysis

We first define

Definition F.4. *For each $k \in [p]$, $l \in \mathbb{R}^+$, we define $\eta_{k,l} := 1 - (1 - p_l)^{b_k}$ to be the probability that at least one element from B_k is sampled with the sampling probability of $p_l = 2^{-l}$.*

Set $\lambda = \Theta(P \log(1/\delta))$. Let $\eta^*_{k,l}$ be the probability that an element from B_k is contained in $H_{1,l}$, such that, for $H_{r,l}$ with any other r, the probability is the same as $\eta^*_{k,l}$.

Lemma F.5 (Sample Probability and Track Probability)**.** *For any layer $k \in [P]$ we have $\eta^*_{k,l} < \eta_{k,l}$. Let δ and ϵ denote the two parameters such that $0 < \delta < \epsilon < 1$. If layer k is a β-important layer and $p_l b_k = O(1)$, then*

$$\eta^*_{k,l} \geq (1 - \Theta(\epsilon)) \cdot \eta_{k,l}.$$

Proof. If one is in $H_{r,l}$, it has to be sampled, so we have $\eta^*_{k,l} \leq \eta_{k,l}$. On the other hand, using Lemma F.3, with probability at least $1 - t\exp(-\Omega(\frac{\lambda p b_k}{t\beta}))$, layer k is $\frac{\beta}{\lambda p b_k}$-trackable.

We have

$$\frac{\beta}{\lambda p b_k} = \Theta(\frac{\beta}{P \log(1/\delta)})$$

where the last step follows from definition of λ.

Thus with probability at least

$$\eta_{k,l}(1 - \Theta(\delta)) \geq \eta_{k,l}(1 - O(\epsilon))$$

an element from B_k is sampled and the element is reported by BATCHHEAVYHITTER.

Thus we complete the proof. □

Lemma F.6 (Probability Approximation)**.** *For $k \in [P]$, let $\widehat{\eta}_k$ be defined as in Line 34 in Algorithm 7. With probability at least $1 - \delta^{\Omega(\log d)}$, for all $k \in [P]$, if $\widehat{\eta}_k \neq 0$ then*

$$(1 - O(\epsilon_1))\eta^*_{k,q_k} \leq \widehat{\eta}_k \leq \eta^*_{k,q_k}.$$

Proof. We define

$$\gamma := \frac{R \log(1/\delta)}{100 \log d}.$$

Recall the condition of Line 34, if $\widehat{\eta}_k \neq 0$, then we have

$$A_{q_k,k} \geq \gamma.$$

For a fixed $k \in [P]$, since the sampling process is independent for each $r \in [R]$, we can assume that

$$\mathbb{E}[A_{q_k,k}] = R\eta_{k,q_k} \geq \gamma.$$

Otherwise, by Chernoff bound (Lemma A.2), we get that

$$\Pr[A_{q_k,k} \geq \gamma] = o(\delta^{\Omega(\log d)}),$$

which implies that with probability at least $1 - \delta^{\Omega(\log d)}$, $\widehat{\eta}_k = 0$ for all $k \in [P]$, and we are done.

Thus, under this assumption, by Chernoff bound (Lemma A.2), we have

$$\Pr[|A_{q_k,k} - R \cdot \eta^*_{k,q_k}| \geq \epsilon R \eta_{k,q_k}] \leq \exp(-\Omega(\epsilon^2 \gamma)) = \delta^{\Omega(\log d)}.$$

Since $P = \mathrm{poly}\log(d)$, by union bound over the B_k , the event

$$(1 - \epsilon_1)\eta_{k,q_k} \leq \frac{A_{q_k,k}}{R_k} \leq (1 + \epsilon_1)\eta_{k,q_k}$$

holds for all $k \in [P]$ with probability at least $1 - \delta^{\Omega(\log(d))}$. Since $\widehat{\eta}_k = \frac{A_{k,q_k}}{R(1+\epsilon_1)}$ by definition, we get the desired bounds.

Thus we complete the proof. □

Definition F.7. *We define q_k as*

$$q_k := \max_{l \in [L]} \Big\{ l | A_{l,k} \geq \frac{R \log(1/\delta)}{100 \log(d)} \Big\}.$$

We say that q_k is well-defined if the set in the RHS of the above definition is non-empty.

At Line 33 in Algorithm 7, we define the q_k^i for i-th vector as the definition above.

Lemma F.8 (Maximizer Probability)**.** *If layer $k \in [P]$ is important (Line 33 in Algorithm 7), then with probability at least $1 - \delta^{\Omega(\log d)}$, the maximizer q_k is well defined.*

Proof. Lemma F.5 tells us that, when $p_k b_k = O(1)$, layer k is at least $\Omega(\frac{\beta}{P \log(1/\delta)})$-trackable. On the other hand, if $P_k = 2^{-l_0} = 1/b_k$, we have $\eta_{k,l_0} = 1 - (1 - p_k)^{b_k} \geq 1/e$. Thus we have

$$\mathbb{E}[A_{l_0,k}] \geq R/e,$$

so by Chernoff bound (Lemma A.2),

$$\begin{aligned} \Pr[A_{l_0,k} \leq \frac{R \log(1/\delta)}{\log d}] &\leq \exp(-\Omega(\frac{R \log(1/\delta)}{\log d})) \\ &\leq \delta^{\Omega(\log d)}. \end{aligned}$$

Thus we show that, there exists one $q_k \geq l_0$ with probability at least $1 - \delta^{\Omega(\log n)}$.

Since there are at most $P = \mathrm{poly} \log(d)$ important layers, with probability at least $1 - \delta^{\Omega(\log n)}$, the corresponding value of q_k is well defined for all important layers.

Thus we complete the proof. □

F.4 From Probability Estimation to Layer Size Approximation

The following lemma shows that, if we have a sharp estimate for the track probability of a layer, then we can obtain a good approximation for its size.

Lemma F.9 (Track probability implies layer size approximation)**.** *Suppose $q_k \geq 1$, $\epsilon \in (0, 1/2)$, and d is sufficiently large. Define $\epsilon_1 := O(\epsilon^2 / \log(d))$. We have for $k \in [P]$,*

- *Part 1. If $\widehat{\eta}_k \leq \eta_{k,q_k}$ then $c_k \leq b_k$.*
- *Part 2. If $\widehat{\eta}_i \geq (1 - \epsilon_1)\eta_{k,q_k}$ then $c_k \geq (1 - O(\epsilon_1))b_k$.*

Proof. We know that

$$b_k = \frac{\log(1 - \eta_{k,q_k})}{\log(1 - 2^{-q_k})},$$

which is a increasing function of η_{k,q_k}.

Part 1. If $\widehat{\eta}_k \leq \eta_{k,q_k}$, then $c_k \leq b_k$.

Part 2. If $\widehat{\eta}_k \geq (1 - O(\epsilon))\eta_{k,q_k}$ we have

$$c_k \geq b_k + \frac{\epsilon_1 \eta_{k,q_k}}{(1 - \eta_{k,q_k}) \log(1 - 2^{-q_k})} \geq b_k - O(\epsilon) b_k.$$

Thus we complete the proof. □

G Space Complexity

In this section, we prove the space complexity of our data structure.

Lemma G.1 (Space complexity of our data structure, formal version of Lemma 4.4)**.** *Our data structure (Algorithm 4 and 5) uses $O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$ space.*

Proof. First, we store the original data,

$$\text{space for } x = O(nd).$$

Second, we store the sub stream/sample of original data

$$\begin{aligned}\text{space for } \overline{x} &= O(RLUnd) \\ &= O(\epsilon^{-4}\log(n/\delta)\log^4(d)\cdot\log(d)\cdot\log(nd^2/\delta)\cdot nd) \\ &= O(\epsilon^{-4}nd\log^6(nd/\delta)).\end{aligned}$$

Our data structure holds a set $\{S_{r,l,u}\}_{r\in[R],l\in[L],u\in[U]}$ (Line 10). Each $S_{r,l,u}$ has a size of $\mathcal{S}_{\mathrm{space}}(\sqrt{\beta}, d) = O(n\cdot\beta^{-1}\log^2(d)\log(nd))$, which uses the space of

$$\begin{aligned}\text{space for } S &= O(RLUn\cdot\beta^{-1}\log^2(d)\log(nd)) \\ &= O(\epsilon^{-4}\log(n/\delta)\log^4(d)\cdot\log(d)\cdot\log(nd)\cdot\log(nd^2/\delta)\cdot n\cdot(\epsilon^{-5}\cdot\mathrm{mmc}(l)^2\log^5(d))\log^2(d)) \\ &= O(\epsilon^{-9}n\cdot\mathrm{mmc}(l)^2\log^{14}(nd/\delta))\end{aligned}$$

where the first step follows from the definition of R, L, U, and the second step follows just simplifying the last step.

We hold a bmap (Line 17 and Line 19) of size $O(RLUd)$, which uses the space of

$$\begin{aligned}\text{space for bmap} &= O(RLUd) \\ &= O(\epsilon^{-4}\log(n/\delta)\log^4(d)\cdot\log(d)\cdot\log(nd^2/\delta)\cdot d) \\ &= O(\epsilon^{-4}d\log^6(nd/\delta)).\end{aligned}$$

In QUERY, we generate a set of sets $\{H_{r,l,u}\}_{r\in[R],l\in[L],u\in[U]}$, each of the sets has size of $O(\beta^{-1})$, so the whole set uses space of

$$\begin{aligned}\text{space for } H &= O(RLU\cdot\beta^{-1}) \\ &= O(\epsilon^{-4}\log(n/\delta)\log^4(d)\cdot\log(d)\cdot\log(nd^2/\delta)\cdot\epsilon^{-5}\cdot\mathrm{mmc}(l)^2\log^5(d)) \\ &= O(\epsilon^{-9}\cdot\mathrm{mmc}(l)^2\log^{11}(nd/\delta)).\end{aligned}$$

By putting them together, we have the total space is

$$\begin{aligned}&\text{total space} \\ &= \text{space for } x + \text{space for } \overline{x} + \text{space for } S + \text{space for bmap} + \text{space for } H \\ &= O(nd) + O(\epsilon^{-4}nd\log^6(nd/\delta)) + O(\epsilon^{-9}n\cdot\mathrm{mmc}(l)^2\log^{13}(nd/\delta)) \\ &\quad + O(\epsilon^{-4}d\log^6(nd/\delta)) + O(\epsilon^{-9}\cdot\mathrm{mmc}(l)^2\log^{11}(nd/\delta)) \\ &= O(\epsilon^{-9}n(d+\mathrm{mmc}(l)^2)\log^{14}(nd/\delta)).\end{aligned}$$

Thus, we complete the proof.

□

H Lower Bound From Previous Work

We first define turnstile streaming model (see page 2 of [LNNT16] as an example) as follows

Definition H.1 (Turnstile streaming model)**.** *We define two different turnstile streaming models here:*

- *Strict turnstile: Each update Δ may be an arbitrary positive or negative number, but we are promised that $x_i \geq 0$ for all $i \in [n]$ at all points in the stream.*
- *General turnstile: Each update Δ may be an arbitrary positive or negative number, and there is no promise that $x_i \geq 0$ always. Entries in x may be negative.*

Under the turnstile model, the *norm estimation problem* has the following streaming lower bound:

Theorem H.2 (Theorem 1.2 in [BBC+17]). *Let l be a symmetric norm on $\mathbb{R}^n$. Any turnstile streaming algorithm (Definition H.1) that outputs, with probability at least 0.99, a $(1 \pm 1/6)$-approximation for $l(\cdot)$ must use $\Omega(\mathrm{mmc}(l)^2)$ bits of space in the worst case.*

We note that this problem is a special case of our symmetry norm distance oracle problem (i.e., with $n = 1$ and query vector $q = \mathbf{0}_d$ where $\mathbf{0}_d$ is a all zeros length-d vector). And in this case, the query time of our data structure becomes $\widetilde{O}(\mathrm{mmc}(l)^2)$ for a constant-approximation, matching the streaming lower bound in Theorem H.2.

I Details About Sparse Recovery Tools

In this section we give an instantiation of the sparse recovery tool we use (Definition D.2) in Algorithms 10 - 12. Although the running times of our data structure are slightly worse (by some log factors) than the result of [KNPW11], it's enough for our symmetric norm estimation task. In terms of space requirement, the classical sparse recovery/compressed sensing only sublinear space is allowed. In our application, we're allowed to use linear space (e.g. d per point, nd in total). And more importantly, our instantiation has much simpler algorithm and analysis than the prior result.

Algorithm 10 Our CountSketch for Batch heavy hitter

data structure BasicBatchHeavyHitter ▷ Definition D.2
members
  $d, n \in \mathbb{N}^+$ ▷ n is the number of vectors, d is the dimension.
  ϵ ▷ We are asking for ϵ-heavy hitters
  δ ▷ δ is the failure probability
  B ▷ B is the multiple number of each counter to take mean
  L ▷ L is the number of number of the level of the binary tree.
  η ▷ η is the precision for l_2 norm estimation.
  K ▷ K is the number of hash functions.
  $\{h_{l,k}\}_{l \in \{0,\ldots,L\}, k \in [K]} \subseteq [2^l] \times [B]$ ▷ hash functions.
  $\{C^i_{l,b}\}_{i \in [n], l \in \{0,\ldots,L\}, b \in [B]}$ ▷ The counters we maintain in CountSketch.
  $\{\sigma_{l,k}\}_{l \in \{0,\ldots,L\}, k \in [K]} \subseteq [d] \times \{+1, -1\}$ ▷ The hash function we use for norm estimation
  $\mathcal{Q}$ ▷ A instantiation of LpLpTailEstimation (Algorithm 13)
  $\{\mathcal{D}_l\}_{l \in [L]}$ ▷ Instantiations of FpEst (Algorithm 14)
end members

public:
procedure Init($\epsilon, n \in \mathbb{N}^+, d \in \mathbb{N}^+, \delta$)
  $L \leftarrow \log_2 d$
  $\delta' \leftarrow \epsilon\delta/(12(\log d) + 1)$
  for $l \in \{0, \ldots, L\}$ **do**
    $\mathcal{D}_l$.Init($n, d, l, 1/7, \epsilon, \delta'$)
  end for
  $C_0 \leftarrow$ greater than 1000
  $\mathcal{Q}$.Init($n, \epsilon^{-2}, 2, C_0, \delta$)
end procedure

The following lemma shows that the sparse recovery data structure satisfies our requirements in Theorem D.3.

I.1 Our Sparse Recovery Tool

We first state the correctness, the proof follows from framework of [KNPW11].

Lemma I.1. *The function* Decode(i, ϵ, d, δ) *(Algorithm 12) returns a set $S \subseteq d$ of size $|S| = O(\epsilon^{-2})$ containing all ϵ-heavy hitters of the i-column of the matrix under l_2 with probability of $1 - \delta$. Here we say j is an ϵ-heavy hitter under l_2 if $|x_j| \geq \epsilon \cdot \|x_{\overline{[\epsilon^{-2}]}}\|_2$ where $x_{\overline{[k]}}$ denotes the vector x with the largest k entries (in absolute value) set to zero. Note that the number of heavy hitters never exceeds $2/\epsilon^2$.*

Algorithm 11 Our CountSketch for Batch heavy hitter

```
data structure BasicBatchHeavyHitter                                  ▷ Definition D.2
procedure EncodeSingle(i ∈ [n], j ∈ [d], z ∈ ℝ, d)
    for l ∈ {0, ..., L} do
        D_l.Update(i, j, z)
    end for
    Q.Update(i, j, z)
end procedure

procedure Encode(i ∈ [n], z ∈ ℝ^d, d)
    for j ∈ [d] do
        EncodeSingle(i, j, z_j, d)
    end for
end procedure

end data structure
```

Algorithm 12 Our CountSketch for Batch heavy hitter

```
data structure BasicBatchHeavyHitter                                  ▷ Definition D.2
procedure Subtract(i, j, k ∈ [n])
    for l ∈ {0, ..., L} do
        D_l.Subtract(i, j, k)
    end for
    Q.Subtract(i, j, k)
end procedure

procedure Decode(i ∈ [n], ε, d)
    EstNorm ← Q.Query(i)    ▷ Here the EstNorm is the estimated tail l_2-norm of i-vector.
    S ← {0}, S' ← ∅
    for l ∈ {0, ..., L} do                                            ▷ The dyadic trick.
        for ξ ∈ S do
            Est ← D.Query(i, ξ)
            if Est ≥ (3/4)ε^2 · EstNorm then
                S' ← S' ∪ {2ξ, 2ξ + 1}
            end if
        end for
        S ← S', S' ← ∅
    end for
    return S
end procedure
end data structure
```

Proof. The correctness follows from the framework of [KNPW11], and combining tail estimation (Lemma I.4) and norm estimation.

□

We next analyze the running time of our data structure in the following lemma:

Lemma I.2. *The time complexity of our data structure (Algorithm 10, Algorithm 11 and Algorithm 12) is as follows:*

- Init *takes time of* $O(\epsilon^{-1}(n+d)\log^2(nd/\delta))$.
- EncodeSingle *takes time of* $O(\log^2(nd/\delta))$.
- Encode *takes time of* $O(d\log^2(nd/\delta))$.
- Subtract *takes time of* $O(\epsilon^{-1}\log^2(nd/\delta))$.

Algorithm 13 Batched ℓ_p tail estimation algorithm, based on [NS19]

data structure LPLPTAILESTIMATION
members
  m ▷ m is the sketch size
  $\{g_{j,t}\}_{j\in[d],t\in[m]}$ ▷ random variable that sampled i.i.d. from distribution $\mathcal{D}_p$
  $\{\delta_{j,t}\}_{j\in[d],t\in[m]}$ ▷ Bernoulli random variable with $\mathbb{E}[\delta_{j,t}] = 1/(100k)$
  $\{y_{i,t}\}_{i\in[n],t\in[m]}$ ▷ Counters
end members
public:
procedure INIT(n, k, p, C_0, δ)
  $m \leftarrow O(\log(n/\delta))$
  Choose $\{g_{j,t}\}_{j\in[d],t\in[m]}$ to be random variable that sampled i.i.d. from distribution $\mathcal{D}_p$
  Choose $\{\delta_{j,t}\}_{j\in[d],t\in[m]}$ to be Bernoulli random variable with $\mathbb{E}[\delta_{j,t}] = 1/(100k)$ ▷ Matrix A in Lemma I.4 is implicitly constructed based on $g_{j,t}$ and $\delta_{j,t}$
  initialize $\{y_{i,t}\}_{i\in[n],t\in[m]} = \{0\}$
end procedure

procedure UPDATE($i \in [n], j \in [d], z \in \mathbb{R}$)
  for $t \in [m]$ **do**
    $y_{i,t} \leftarrow y_{i,t} + \delta_{j,t} \cdot g_{jt} \cdot z$
  end for
end procedure

procedure SUBTRACT($i, j, k \in [n]$)
  for $t \in [m]$ **do**
    $y_{i,t} \leftarrow y_{j,t} - y_{k,t}$
  end for
end procedure

procedure QUERY($i \in [n]$)
  $V \leftarrow \mathrm{median}_{t\in[m]} |y_{i,t}|^2$
  return V
end procedure

- DECODE *takes time of* $O(\epsilon^{-2}\log^2(nd/\delta))$.

Proof. We first notice that $L = \log_2 d$ and $\delta' = \epsilon\delta/(12(\log(d)+1))$.

For the procedure INIT (Algorithm 10), Line 22 takes time

$$O(L\epsilon^{-1}n\log(n/\delta')) = O(\epsilon^{-1}n\log^2(\delta^{-1}\epsilon^{-1}nd\log(d)).$$

Line 25 takes time $O((n+d)\log(n/\delta))$. Taking together, we have the total running time of INIT is $O(\epsilon^{-1}(n+d)\log^2(\delta^{-1}\epsilon^{-1}nd\log(d))$

For the procedure ENCODESINGLE (Algorithm 11), Line 4 takes time of

$$O(L\log(n/\delta')) = O(\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

Line 6 takes time of $O(\log(n/\delta))$. Taking together we have the total running time of ENCODESINGLE to be $O(\log^2(\epsilon^{-1}\delta^{-1}nd\log(d)))$.

For the procedure ENCODE (Algorithm 11), it runs ENCODESINGLE for d times, so its running time is $O(d\log^2(\epsilon^{-1}\delta^{-1}nd\log(d)))$.

For the procedure SUBTRACT (Algorithm 12), Line 4 runs in time

$$O(L\epsilon^{-1}\log(n/\delta')) = O(\epsilon^{-1}\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

Line 6 runs in time $O(\log(n/\delta))$. So the total running time is $O(\epsilon^{-1}\log^2(\epsilon^{-1}\delta^{-1}nd\log(d)))$.

For the procedure DECODE (Algorithm 12), Line 10 runs in time $O(\log(n/\delta))$. Line 14 runs in time

$$O(L\epsilon^{-2}\log(n/\delta')) = O(\epsilon^{-2}\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

So the total running time is $O(\epsilon^{-2}\log^2(\epsilon^{-1}\delta^{-1}nd\log(d)))$

Thus we complete the proof. □

The space complexity of our data structure is stated in below.

Lemma I.3. *Our batch heavy hitter data structure (Algorithm 10, Algorithm 11 and Algorithm 12) takes the space of*

$$O(\epsilon^{-1}(n+d)\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

Proof. Our data structure has these two parts to be considered:

- The instantiations of FPEST we maintain.
- The instantiation of LPLPTAILESTIMATION we maintain .

The first part takes the space of

$$O(L\epsilon^{-1}n\log(n/\delta')) = O(\epsilon^{-1}n\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

And the second part takes the space of $O(d\log(n/\delta))$. Adding them together we complete the proof. □

I.2 Lp Tail Estimation

[NS19] provide a linear data structure LPLPTAILESTIMATION(x, k, p, C_0, δ) that can output the estimation of the contribution of non-heavy-hitter entries. We restate their Lemma as followed.

Lemma I.4 (Lemma C.4 of [NS19])**.** *Let $C_0 \geq 1000$ denote some fixed constant. There is an oblivious construction of matrix $A \in \mathbb{R}^{m\times n}$ with $m = O(\log(1/\delta))$ along with a decoding procedure* LPLPTAILESTIMATION(x, k, p, C_0, δ) *such that, given Ax, it is possible to output a value V in time O(m) such that*

$$\frac{1}{10k}\|x_{\overline{C_0 k}}\|_p^p \leq V \leq \frac{1}{k}\|x_{\overline{k}}\|_p^p,$$

holds with probability $1-\delta$.

Lemma I.5. *The running time of the above data structure is*

- INIT *runs in time of* $O((n+d)\log(n/\delta))$
- UPDATE *runs in time of* $O(\log(n/\delta))$
- SUBTRACT *runs in time of* $O(\log(n/\delta))$
- QUERY *runs in time of* $O(\log(n/\delta))$

And its space is $O(d\log(n/\delta))$.

I.3 Lp Norm Estimation

Following the work of [KNW10], we provide a linear sketch satisfying the following requirements.

Lemma I.6 ([KNW10])**.** *There is an linear sketch data structure* FPEST *using space of*

$$O(\epsilon^{-1}\phi^{-2}n\log(n/\delta))$$

and it provide these functions:

- INIT$(n \in \mathbb{Z}^+, d \in \mathbb{Z}^+, l \in \mathbb{Z}^+, \phi, \epsilon, \delta)$*: Initialize the sketches, running in time* $O(\epsilon^{-1}n\log(n/\delta))$

Algorithm 14 Batched ℓ_p norm estimation algorithm, based on [KNW10]

data structure LPNORMEST
members
  R, T ▷ parallel parameters
  m ▷ Sketch size
  $\{y^i_{r,t}\}_{i\in[n],r\in[R],t\in[T]} \subset \mathbb{R}^m$ ▷ Sketch vectors
  $\{A\}_{r,t} \subset \mathbb{R}^{m\times 2^l}$ ▷ Sketch matrices
  $\{h_t\}_{t\in[T]} : \{0,\ldots,2^l\} \to [R]$ ▷ hash functions
end members

public:
procedure INIT$(n, d, l, \phi, \epsilon)$
  $R \leftarrow \lceil 1/\epsilon \rceil$
  $T \leftarrow \Theta(\log(n/\delta))$
  $m \leftarrow O(1/\phi^2)$
  for $i \in [n], r \in [R], t \in [T]$ **do**
    $y^i_{r,t} \leftarrow \mathbf{0}$ ▷ Sketch vectors
  end for
  for $i \in [n], r \in [R], t \in [T]$ **do**
    generate $A_{r,t} \in \mathbb{R}^{m\times 2^l}$ ▷ See [KNW10] for details
  end for
  initialize h
end procedure

procedure UPDATE$(i \in [n], j \in [d], z \in \mathbb{R})$
  $\xi \leftarrow j \cdot 2^l/d$
  for $t \in [T]$ **do**
    $y^i_{h_t(\xi),t} \leftarrow y^i_{h_t(\xi),t} + A_{h_t(\xi),t}\mathbf{i}^l_{j,z}$ ▷ $\mathbf{i}^l_{j,z}$ is define to be the 2^l-dimensional vector with j-th entry of z and others to be 0
  end for
end procedure

procedure SUBTRACT$(i, j, k \in [n])$
  for $r \in [R], t \in [T]$ **do**
    $y^i_{r,t} \leftarrow y^j_{r,t} - y^k_{r,t}$
  end for
end procedure

procedure QUERY$(i \in [n], \xi \in [2^l])$
  for $t \in [T]$ **do**
    $V_{h_t(\xi),t} \leftarrow \mathrm{median}_{\zeta\in[m]}\, y^i_{r,t,\zeta} / \mathrm{median}(|\mathcal{D}_p|)$ ▷ Definition A.5
  end for
  $V \leftarrow \mathrm{median}_{t\in[T]}\, V_{h_t(\xi),t}$
  return V
end procedure

- UPDATE$(i \in [n], j \in [d], z \in \mathbb{R})$*: Update the sketches, running in time* $O(\log(n/\delta))$
- SUBTRACT$(i, j, k \in [n])$*: Subtract the sketches, running in time* $O(\epsilon^{-1}\phi^{-2}\log(n/\delta))$
- QUERY$(i \in [n], \xi \in [2^l])$*: This function will output a* V *satisfying*

$$(1-\phi)\cdot F_i(l,\xi) \le V \le (1+\phi)\cdot(F_i(l,\xi) + 5\epsilon\|x_i\|_2^2),$$

where $F(l, \xi)$ is defined as

$$F_i(l, \xi) := \sum_{j=\frac{\xi}{2^l}d}^{\frac{\xi+1}{2^l}d-1} |x_{i,j}|^2,$$

running in time $O(\phi^{-2}\log(n/\delta))$

We choose ϕ to be constant when we use the above Lemma.