On the Robustness of CountSketch to Adaptive Inputs

Edith Cohen [1 2] Xin Lyu [3] Jelani Nelson [3 1] Tamás Sarlós [1] Moshe Shechner [2] Uri Stemmer [2 1]

Abstract

The last decade saw impressive progress towards understanding the performance of algorithms in *adaptive* settings, where subsequent inputs may depend on the output from prior inputs. Adaptive settings arise in processes with feedback or with adversarial attacks. Existing designs of robust algorithms are generic wrappers of non-robust counterparts and leave open the possibility of better tailored designs. The lowers bounds (attacks) are similarly worst-case and their significance to practical setting is unclear. Aiming to understand these questions, we study the robustness of `CountSketch`, a popular dimensionality reduction technique that maps vectors to a lower dimension using randomized linear measurements. The sketch supports recovering ℓ_2-heavy hitters of a vector (entries with $v[i]^2 \geq \frac{1}{k}\|\boldsymbol{v}\|_2^2$). We show that the classic estimator is not robust, and can be attacked with a number of queries of the order of the sketch size. We propose a robust estimator (for a slightly modified sketch) that allows for quadratic number of queries in the sketch size, which is an improvement factor of $\sqrt{k}$ (for k heavy hitters) over prior "blackbox" approaches.

1. Introduction

Algorithms are often analyzed under the assumption that their internal randomness is independent of their inputs. This assumption, however, is not always reasonable. For example, consider a large system where there is a feedback loop between inputs and outputs or an explicit adversarial attack aimed at constructing inputs on which the system fails. In such a case, it is no longer true that the inputs are generated independently of the algorithm's randomness (as the inputs depend on previous outputs which depend on the internal randomness) and hence the algorithms might fail to provide utility.

This motivated a growing interest in understanding the performance of algorithms when the inputs are chosen *adaptively*, possibly as a function of their previous outputs, and in designing *robust algorithms* which provide utility guarantees even when their inputs are chosen adaptively. Works in this vein span multiple areas, including machine learning (48; 24; 5; 43), adaptive data analysis (22; 31; 37; 28; 19), dynamic graph algorithms (46; 3; 23; 27; 49; 9), and sketching and streaming algorithms (39; 29; 12; 30; 51; 6; 11). However, the resulting (robust) algorithms tend to be significantly less efficient then their classical counterparts. This naturally raises the question of quantifying more precisely the robustness of algorithms. We continue the study of this question for sketching algorithms (though our techniques are applicable more broadly).

The study of sketching in adversarial environments was initiated by (39), in a setting different from ours in which the adversary fully knows the public randomness when designing adversarial inputs in a multi-party setting. Robustness of sketching algorithms to adaptive inputs (in a setting similar to ours) was then considered implicitly by (3), who showed an impossibility result for robust L_0 sampling in sublinear memory. Adaptive robustness in sketching was then formalized and put forward explicitly by (29), who showed general impossibility results for linear sketches. A recent line of work showed *positive results* (i.e., *robust algorithms*) for many problems of interest, starting with (12), and continuing with (30; 51; 6; 11). However, the resulting robust algorithms are generally significantly less efficient then their classical counterparts. Furthermore, while the analyses of classical algorithms do not seem to carry over to the robust setting, so far we do not have any example showing that they are not robust (i.e,. for all we know it might only be the analysis that breaks).[1] In fact, so far we do not have any

[1]Google Research [2]Tel Aviv University [3]UC Berkeley. Correspondence to: <`edith@cohenwang.com`, `lyuxin1999@gmail.com`, `minilek@alum.mit.edu`, `stamas@google.com`, `moshe.shechner@gmail.com`, `u@uri.co.il`>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

[1]As we mentioned, (29) showed negative results for linear sketches in the adaptive setting; however, the negative results are very far from respective upper bounds. It is also worth mentioning that (12) showed an attack on a simplified version of the classical AMS sketch (4) (with a weaker estimator). However, their attack does not apply with the classic estimator.

example of a classical sketching algorithm that fails to be robust (as is). We believe that understanding the extent to which classical algorithms are (or fail to be) robust is crucial for justifying the work on robust algorithms.

Driven by the above discussion, in this work we set out to explore the robustness properties of the classical `CountSketch` algorithm (15), related to *feature hashing* (41) in the machine learning literature. `CountSketch` is a popular dimensionality reduction technique that maps vectors to a lower-dimension using randomized linear measurements. The method is textbook material and has found many applications in machine learning and data analysis (50; 45; 16; 17; 1; 47; 2; 18). The sketch is often a component of large ML models or data analytics systems and its robustness may impact the overall robustness of the system. In some applications there is an explicit feedback loop between inputs and outputs (47; 44), where the output of the sketch is used to update the parameters of a machine learning model in each training step, which then determines the next input.

Operationally, `CountSketch` is parametrized by (n, d, b), where n is the dimension of input vectors, d is the size of the sketch, and b is a parameter controlling the accuracy of the sketch (referred to as the "width" of the sketch). It is applied by initializing d/b pairs of random hash functions $\boldsymbol{\rho} = \big((h_1, s_1), \ldots, (h_{d/b}, s_{d/b})\big)$ where $h_j : [n] \to [b]$ and $s_j : [n] \to \{\pm 1\}$. We think of $\boldsymbol{\rho}$ as defining $\frac{d}{b} \times b$ "buckets" (b buckets for every pair of hash functions). To sketch a vector $\boldsymbol{v} \in \mathbb{R}^n$: For every $i \in [n]$ and for every $j \in [d/b]$, add $s_j(i) \cdot v[i]$ to the bucket indexed by $(j, h_j(i))$. The resulting collection of d summations (the values of the buckets) is the sketch, which we denote as $\texttt{Sketch}_{\boldsymbol{\rho}}(\boldsymbol{v})$. That is, $\texttt{Sketch}_{\boldsymbol{\rho}}(\boldsymbol{v}) := \big(c_{(j,w)}\big)_{j \in [d/b], w \in [b]}$, where

$$c_{j,w} := \sum_{i \in [n] : h_j(i) = w} s_j(i) \cdot v[i].$$

In applications, the desired task (e.g., recovering the set of heavy hitter entries of $\boldsymbol{v}$ and approximate values, reconstructing an approximation $\hat{\boldsymbol{v}}$ of the input vector, or approximating the inner product of two vectors) is obtained by applying an *estimator* M to the sketch. Note that M does not access the original vector. The most commonly used estimator in the context of `CountSketch` is the *median estimator*, with which the ℓth enrty of the original vector is estimated as $\text{Median}_{j \in [d/b]} \left\{ s_j(\ell) \cdot c_{(j, h_j(\ell))} \right\}$. To intuit this estimator, observe that for every $j \in [d/b]$ we have that

$$\begin{aligned} s_j(\ell) \cdot c_{\left(j, h_j(\ell)\right)} &= s_j(\ell) \cdot \left(\sum_{i \in [n]:\ h_j(i) = h_j(\ell)} s_j(i) \cdot v[i] \right) \\ &= \boldsymbol{v}[\ell] + \left(\sum_{i \neq \ell:\ h_j(i) = h_j(\ell)} s_j(\ell) \cdot s_j(i) \cdot v[i] \right) \end{aligned}$$

is an unbiased estimator for $\boldsymbol{v}[\ell]$. Hence, intuitively, the median of these values is a good estimate.

Our focus will be on the task of recovering the ℓ_2-heavy hitters. An ℓ_2-heavy hitter with parameter k of a vector $\boldsymbol{v}$ is defined to be an entry i such that $\boldsymbol{v}[i]^2 > \frac{1}{k} \|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2^2$, where $\boldsymbol{v}_{\mathsf{tail}[k]}$, the k-tail of $\boldsymbol{v}$, is a vector obtained from $\boldsymbol{v}$ by replacing its k largest entries in magnitude with 0. The *heavy-hitters problem* is to return a set of $O(k)$ keys that includes all heavy hitters. The output is correct when all heavy hitters are reported. In the non-adaptive setting, this problem can be solved using `CountSketch` with $d = O(k \log n)$ and $b = O(k)$, by returning the $O(k)$ keys with the largest estimated magnitudes (via the median estimator).

As we mentioned, in this work we are interested in the *adaptive setting*, where the same initialization (specified by $\boldsymbol{\rho}$) is used to sketch different inputs or to maintain a sketch as the input is updated. In its most basic form, the setting is modeled using the following game between a sketching algorithm `Sketch` with an estimator M (not necessarily `CountSketch` and the median estimator) and an `Analyst`. At the beginning, we sample the initialization randomness of `Sketch`, denoted as $\boldsymbol{\rho}$. Then, the game proceeds in m rounds, where in round $q \in [m]$:

- The `Analyst` chooses a vector $\boldsymbol{v}_q \in \mathbb{R}^n$, which can depend, in particular, on all previous outputs of M.
- $M(\texttt{Sketch}_{\boldsymbol{\rho}}(\boldsymbol{v}_q))$ outputs a set K_q of $O(k)$ keys, which is given to the `Analyst`.

We say that $(\texttt{Sketch}, M)$ is *robust for m rounds* if for any `Analyst`, with high probability, for every $q \in [m]$ it holds that K_q contains all the heavy hitters of $\boldsymbol{v}_q$. The focus is on designing robust sketches of size as small as possible (as a function of n, k, m). We remark that it is trivial to design robust sketches with size linear with m, by simply duplicating a classical sketch m times and using every copy of the classical sketch to answer one query. Therefore, if a sketch is only robust to a number of rounds that scales linearly with its size, then we simply say that this sketch is *non-robust*.

1.1. Our Contributions

We first briefly state our main contributions. We elaborate on these afterwards.

1. We show that `CountSketch` together with the standard median estimator are non-robust. We achieve this by designing an attack that poses $O(d/b)$ queries to `CountSketch` such that, w.h.p., the answer given to the last query is wrong. This constitutes the first result showing that a popular sketching algorithm is non-

robust. We complement our analysis with empirical evaluation, showing that our attack is practical.

2. We introduce (see Section 3) a novel estimator (instead of the median estimator), which we refer to as the *sign-alignment estimator*, that reports keys as heavy hitters based on the *signs* of their corresponding buckets. This new estimator is natural, and as we show, has comparable performances to the median estimator even in the non-adaptive setting. We believe that this new estimator can be of independent interest.

3. We design a noisy version of our sign-alignment estimator (utilizing techniques from differential privacy), and design a variant of `CountSketch`, which we call `BucketCountSketch`, or `BCountSketch` in short. We show that `BCountSketch` together with our noisy estimator are robust for m rounds using sketch size roughly $\approx \sqrt{m} \cdot k$. This improves over the previous state-of-the-art robust algorithm for the heavy hitters problem of (30), which has size $\approx \sqrt{m} \cdot k^{1.5}$.

   We show that, in a sense, the additional ingredients from differential privacy in the robust estimator are necessary: `BCountSketch` and `CountSketch` with a basic version of the sign-alignment estimator are non-robust. Moreover, our analysis of the robust noisy version is tight in that it can be attacked using $\widetilde{O}(m^2)$ rounds.

4. Extensions: We show that our algorithms allow for robustly reporting estimates for the weight of the identified heavy-hitters. We also refine our robust algorithms so that robustness is guaranteed for longer input sequences in some situations. In particular, this is beneficial in a streaming setting, where the input vector changes gradually and we can account for changes in the output only.

1.1.1. Our attack on CountSketch with the median estimator

We next provide a simplified description of our attack on `CountSketch` with the median estimator. We emphasize that our attack is much more general, applicable to a broader family of sketches and estimators (see Appendix E). Recall that on every iteration, when given a vector $\boldsymbol{v}$, we have that $M(\texttt{Sketch}_{\boldsymbol{\rho}}(\boldsymbol{v}))$ outputs a set of $k' = O(k)$ keys. In our attack, we pose a sequence of $m = O(d/b)$ vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_m$, where all of these vectors contain keys 1 and 2 as "largish heavy hitters" (of equal value), as well as a fixed set of $k' - 1$ "super heavy hitters", say keys $3, 4, \dots, k' + 1$. In addition, each $\boldsymbol{v}_q$ contains a disjoint set of keys (say T keys) with random $\{\pm 1\}$ values, which we refer to as a *random tail*. So each of these vectors contains $k' - 1$ "super heavy hitters", two "largish heavy hitters", and random noise. We feed each of these vectors to `CountSketch`. As `CountSketch` reports exactly k' keys, in every iteration we expect all of the "super heavy" elements to be reported, together with *one* of the "largish heavy hitters" (as a function of the random tail). Let us denote by $\text{Collected} \subseteq [m]$ the subset of all iterations during which key 1 was *not* reported. (We refer to a tail used in iteration $q \in \text{Collected}$ as a *collected tail*.) The fact that key 2 was reported over key 1 during these iterations means that our random tails introduced some bias to the weight estimates of key 2 over the weight estimates of key 1. We show that, conditioned on a tail being collected, in expectation, the random tail introduces *negative* bias to the weight estimate of key 1, and *positive* bias to the weight estimate of key 2. At the end of this process, we construct a final query vector $\boldsymbol{v}_{\text{final}}$ containing key 1 as a "super heavy hitter" and keys $2, 3, \dots, k' + 1$ as "largish heavy hitters", together with the sum of all the collected tails. We show that, w.h.p., the biases we generate "add up", such that key 1 *will not get reported as heavy* when querying $\boldsymbol{v}_{\text{final}}$, despite being the dominant coordinate in $\boldsymbol{v}_{\text{final}}$ by a large margin.

Remark 1.1. *An important feature of our attack is that it works even when `CountSketch` is only used to report a set of heavy keys,* without their estimated values. *The attack can be somewhat simplified in case estimated values are reported.*

Remark 1.2. *A natural question is robustness with routine applications of `CountSketch`. We point out that elements of the attack can occur in routine settings. Our attack is "blackbox," only uses information on outputs. The inputs are "natural" with the same heavy keys and random noise. The final query $\boldsymbol{v}_{\text{final}}$ on which the sketch fails is obtained by combining inputs with the same reported heavy key and is the only one that depends on prior inputs. In practice a heavy key can correspond to examples with the same label or to related traffic patterns that might load the same network component. This suggests that an adversarial input can be constructed after simply monitoring the "normal operation" of the algorithm on unintentional "borderline" inputs. Alternatively, a natural feedback process can combine inputs with the same reported heavy key.*

We complement our theoretical analysis of this attack with empirical results, showing that the attack is feasible with a small number of queries. Figure 1 reports simulation results of the attack on the median estimator.[2] The left plot shows the bias-to-noise ratio (the median value of the tail contributions scaled by its standard deviation) as a function of the number of attack rounds for the two special keys (which accumulate positive and negative bias) and another key (which remains unbiased). The left plot visualizes the square-root

[2]Code for all the experiments is available at `https://github.com/google-research/google-research/tree/master/robust_count_sketch`.

relation of the bias-to-noise ratio with the number of rounds. A sketch provides good estimates for a key when its weight is larger than the "noise" on its buckets that is induced by the "tail" of the vector. In this case, the key is a heavy hitter. The attack is thus effective when the bias exceeds the noise. The right plot shows the number of rounds needed to achieve a specified bias-to-noise ratio (showing $\{1, 4\}$) as a function of the size (number of rows $\ell = d/b$) of the sketch. The results indicate that $r \approx 5 \cdot \texttt{BNR}^2 \cdot (d/b)$ rounds are needed to obtain a vector with bias-to-noise ratio of $\texttt{BNR}$ for a sketch with (d/b) rows.[3]

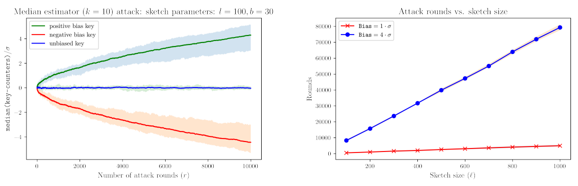

Figure 1. Left: Bias-to-noise ratio for number of rounds, average of 10 simulations with different initializations (shaded region between the minimum and maximum). Right: Attack rounds versus sketch size $\ell = d/b$ to obtain bias-to-noise ratio $\in \{1, 4\}$, averaged over 20 and 5 simulations respectively (shaded region between minimum and maximum).

1.1.2. Our new robust sketch using differential privacy

Differential privacy (20) is a mathematical definition for privacy that aims to enable statistical analyses of datasets while providing strong guarantees that individual level information does not leak. Specifically, an algorithm that analyzes data is said to be *differentially private* if it is insensitive to the addition/deletion of any single individual record in the data. Intuitively, this guarantees that whatever is learned about an individual record could also be learned without this record. Over the last few years, differential privacy has proven to be an important algorithmic notion, even when data privacy is not of concern. Particularly of interest to us is a recent line of work, starting from (19), showing that differential privacy can be used as a tool for avoiding overfitting, which occurs when algorithms begin to "memorize" data rather than "learning" to generalize from a trend.

Recall that the difficulty in our adaptive setting arises from potential dependencies between the inputs of the algorithm and its internal randomness. Our construction builds on a technique, introduced by (30), for using differential privacy to protect not the input data, but rather the internal randomness of algorithm. As (30) showed, leveraging the "anti-overfitting" guarantees of differential privacy, this overcomes the difficulties that arise in the adaptive setting. Following (30), this technique was also used by (6; 11; 25; 9) for designing robust algorithms in various settings. At a high level, all of these constructions operate as follows. Let $\mathcal{A}$ be a randomized algorithm that solves some task of interest *in the non-adaptive setting* (in our case, $\mathcal{A}$ could be `CountSketch` combined with the median estimator). To design a robust version of $\mathcal{A}$ do the following.

```
GenericTemplate

1. Instantiate $P$ copies of the non-adaptive algorithm $\mathcal{A}$
   (for some parameter $P$).
2. For $m$ steps:
   (a) Feed the current input to all of the copies of $\mathcal{A}$ to
       obtain $P$ intermediate outputs $y_1, \dots, y_P$.
   (b) Return a differentially private aggregation of
       $\{y_1, \dots, y_P\}$ as the current output.
```

That is, all the current applications of differential privacy for constructing robust algorithms operate in a black-box fashion, by aggregating the outcomes of non-robust algorithms. For the ℓ_2-heavy hitters problem, as we next explain, we would need to set $P \approx \sqrt{m \cdot k}$, which introduces a blowup of $\approx \sqrt{m \cdot k}$ on top of the size of the non-adaptive algorithm. Applying this construction with `CountSketch`, which has size $\tilde{O}(k)$, results in a robust algorithm for m rounds with size $\approx \sqrt{m} \cdot k^{1.5}$.

The reason for setting $P \approx \sqrt{m \cdot k}$ comes from known composition theorems for differential privacy, showing that, informally, we can release T aggregated outputs in a differentially private manner given a dataset containing $\approx \sqrt{T}$ elements. In our context, we have m rounds, during each of which we need to release $O(k)$ heavy elements. This amounts to a total of $O(m \cdot k)$ aggregations, for which we need to have $\approx \sqrt{m \cdot k}$ intermediate outputs.

We design better algorithms by breaking free from the black-box approach outlined above. We still use differential privacy to protect the internal randomness of our algorithms, but we do so in a "white-box" manner, by integrating it into the sketch itself. As a warmup, consider the following noisy variant of the median estimator (to be applied on top of `CountSketch`). Given the sketch of a vector $\boldsymbol{v}$, denoted as $\texttt{Sketch}_{\boldsymbol{\rho}}(\boldsymbol{v}) := \left(c_{(j,w)}\right)_{j\in[d/b],w\in[b]}$, and a coordinate ℓ, instead of returning the actual median of $\left\{s_j(\ell) \cdot c_{(j,h_j(\ell))}\right\}_{j\in[d/b]}$, return a differentially private estimate for it.

As before, in order to release $O(m \cdot k)$ estimates throughout the execution, we would need to set $d/b \approx \sqrt{m \cdot k}$, which

[3]We performed additional simulations (not shown) where we swept b from 30 to 300, keeping $d/b = 100$ and keeping $k'/b = 3$. We observed (as expected) the same dependence of r. Note that the dependence should hold as long as the parameters are in a regime where most buckets of each of the k' heaviest keys have no collisions with other heavy keys. Also note that we used a large value of $k' = b/3$, but the same dependence holds for smaller values of k'.

results in a sketch of size $\frac{d}{b} \times b \approx \sqrt{m} \cdot k^{1.5}$. So we did not gain much with this idea in terms of the sketch size compared to the `GenericTemplate`. Still, there is a conceptual improvement here. The improvement is that with the `GenericTemplate` we argued about differential privacy w.r.t. the intermediate outputs $y_1, \ldots, y_P$, where every y_p results from a different instantiation of `CountSketch`. This effectively means that in the `GenericTemplate` we needed to tune our privacy parameters so that the aggregation in Step 2b "hides" an entire copy of `CountSketch`, which includes $2d/b$ hash functions. With our warmup idea, on the other hand, when privately estimating the median of $\left\{s_j(\ell) \cdot c_{(j,h_j(\ell))}\right\}_{j\in[d/b]}$, we only need to hide every single one of these elements, which amounts to hiding only a single hash function pair (h_j, s_j). (We refer to the type of object we need to "hide" with differential privacy as our *privacy unit*; so in the `GenericTemplate` the privacy unit was a complete copy of `CountSketch`, and now the privacy unit is reduced to a single hash function pair). This is good because, generally speaking, protecting less with differential privacy is easier, and can potentially be done more efficiently.

Indeed, we obtain our positive results by "lowering the privacy bar" even further. Informally, we show that it is possible to work with the individual *buckets* as our privacy unit, rather than the individual hash functions, while still being able to leverage the generalization properties of differential privacy to argue utility in the adaptive setting. Intuitively, but inaccurately, by doing so we will have $\approx d$ elements to aggregate with differential privacy (the number of buckets), rather than only $\approx d/b \approx d/k$ elements (number of hash functions), which would allow us to answer a larger number of adaptive queries via composition arguments.

There are two major challenges with this approach, which we need to address.

First challenge. Recall that every key i participates in d/b buckets (one for every hash function). So, even if we work with the individual buckets as the privacy unit, still, when estimating the weight of key i we have only d/b elements to aggregate; not d elements as we described it above. It is therefore not clear how to gain from working with the buckets as the privacy unit. We tackle this by conducting a more fine-tuned analysis, based on the following idea. Suppose that the current input vector is $\boldsymbol{v}$ and let H denote the set of $O(k)$ keys identified to be heavy. While we indeed estimate the weight of every $i \in H$ by aggregating only d/b buckets, what we show is that, on average, we need to aggregate *different* buckets to estimate the weight of every $i \in H$. This means that (on average) every bucket participates in very few estimations per query. Overall, every bucket participates in $O(m)$ aggregations throughout the execution, rather than $O(m{\cdot}k)$ as before. Using composition arguments, we now need to aggregate only $\sqrt{m}$ elements (buckets) to produce our estimates, rather than $\sqrt{m \cdot k}$ as before. So it suffices to set $d/b \gtrsim \sqrt{m}$, i.e., suffices to set $d \gtrsim \sqrt{m} \cdot b \approx \sqrt{m} \cdot k$. The analysis of this idea is delicate, and we actually are not aware of a variant of the median estimator that would make this idea go through. To overcome this issue, we propose a novel estimator, which we refer to as the *sign-alignment estimator*, that reports keys as heavy hitters based on the signs of their corresponding buckets. This estimator has several appealing properties that, we believe, make it of independent interest.

Second challenge. The standard generalization properties of differential privacy, which we leverage to avoid the difficulties that arise from adaptivity, only hold for product distributions (Existing works with other distributions (7; 35) do not apply in our setting.) This is fine when working with the individual hash functions as the privacy unit, because the different hash functions are sampled independently. However, this is no longer true when working with the individual buckets as the privacy unit, as clearly, buckets pertaining to the same hash function are dependent. To overcome this difficulty, we propose a variant of `CountSketch` which we call `BCountSketch`, that has the property that all d of its buckets are independent. This variant retains the marginal distribution of the buckets in `CountSketch`, but removes dependencies.

1.1.3. Feasibility of alternatives

Deterministic algorithms are inherently robust and therefore one approach to achieve robustness is to redesign the algorithm to be deterministic or have deterministic components (46; 26). We note that the related ℓ_1-heavy hitters problem on data streams has known deterministic sketches (40; 42), and therefore a sketch that is fully robust in adaptive settings. For ℓ_2-heavy hitters, however, all known designs are based on randomized linear measurements and there are known lower bounds of $\Omega(\sqrt{n})$ on the size of any deterministic sketch, even one that only supports positive updates (32). In particular this means that for ℓ_2-heavy hitters we can not hope for better robustness via a deterministic sketch.

1.1.4. Roadmap

In Section 2 we review `CountSketch` and the classic median estimator and introduce our variant `BCountSketch`. In Section 3 we introduce our sign-alignment estimators: A simple threshold estimator and a stable estimator that is designed for continuous reporting such as in streaming. In Appendix F we report results of an empirical evaluation that demonstrates that these novel sketch/estimators variants perform similarly to classic ones. Our robust estimators (for the threshold and stable versions) are introduced in Section 4 with proof details provided in Appendix A (for the threshold estimator) and Appendix D (for the stable estimator). In Appendix B we extend our robust estimators to include weight estimates for reported keys. In Appendix C we tighten the efficiency of the robust estimators. In Section E we provide details on our attack strategy.

2. Preliminaries

For vectors $\boldsymbol{v}, \boldsymbol{u} \in \mathbb{R}^n$ we use the notation $v[i]$ for the value of the ith entry of the vector (which we also refer to as the ith key), $\langle \boldsymbol{v}, \boldsymbol{u} \rangle = \sum_{i=1}^n v[i]u[i]$ for the inner product, and $\|\boldsymbol{v}\|_2 := (\sum_{i=1}^n v[i]^2)^{1/2}$ for the ℓ_2 norm.

Definition 2.1. *(heavy hitter) Given a vector $\boldsymbol{v} \in \mathbb{R}^n$, an entry $i \in [n]$ is an ℓ_2-k-heavy hitter if $v[i]^2 > \frac{1}{k}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2^2$.*

Definition 2.2. *(Heavy hitters problem, with and without values) A set of entries $K \subset [n]$ is a correct solution for the heavy hitters problem if $|K| = O(k)$ and K includes all the heavy hitters. The solution is α-correct for the problem with values if it includes approximate values $\hat{v}[i]$ for all $i \in K$ so that $|\hat{v}[i] - v[i]| \le (\sqrt{\alpha/k})\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2$.*

2.1. `CountSketch` and `BCountSketch`

`CountSketch` and our proposed variant `BCountSketch` are specified by the parameters (n, d, b), where n is the dimension of input vectors, b is its *width*, and d is the size of the sketch (number of linear measurements of the input vector).

The internal randomness ρ of the sketch specifies a set of d measurement vectors $(\boldsymbol{\mu}_t)_{t\in[d]}$ where $\boldsymbol{\mu}_t \in \{-1, 0, 1\}^n$ for $t \in [d]$. The sketch of a vector $\boldsymbol{v} \in \mathbb{R}^n$ is the set of d linear measurements (which we also refer to as *buckets*)

$$\mathtt{Sketch}_{\boldsymbol{\rho}}(\boldsymbol{v}) := (c_t(\boldsymbol{v}) := \langle \boldsymbol{\mu}_t, \boldsymbol{v} \rangle)_{t\in[d]}\,.$$

`CountSketch`. The internal randomness specifies a set of random hash functions $h_r : [n] \to [b]$ ($r \in [d/b]$) with the marginals that $\forall j \in [b]$, $i \in [n]$, $\mathsf{Pr}[h_r(i) = j] = 1/b$, and $s_r : [n] \to \{-1, 1\}$ ($r \in [d/b]$) so that $\mathsf{Pr}[s_r(i) = 1] = 1/2$. The d measurement vectors are organized as d/b sets of b measurements each. $\boldsymbol{\mu}_{(r-1)\cdot b+j}$ ($r \in [d/b], j \in [b]$):

$$\mu_{(r-1)\cdot b+j}[i] := \mathbb{1}_{h_r(i)=j}\, s_r(i).$$

Interestingly, limited (pairwise) independence of the hash functions h_r and s_r suffices for the utility guarantees (stated below). Note that with `CountSketch` the measurement vectors within each set are dependent.

`BCountSketch`. The measurement vectors are drawn i.i.d. from a distribution $\boldsymbol{\mu} \sim \mathcal{B}$. The distribution $\mathcal{B}$ is the same as that of the measurement vectors of `CountSketch` except that dependencies are removed. The i.i.d. property will facilitate our analysis of robust estimation.

Each $\boldsymbol{\mu}_t$ ($t \in [d]$) is specified by two objects: A selection hash function h_t and a sign hash function s_t with the following marginals:

- $h_t : [n] \to \{0, 1\}$ s.t. $\forall i \in [n] \mathsf{Pr}[h_t(i) = 1] = 1/b$.
- $s_t : [n] \to \{+1, -1\}$ s.t. $\forall i \in [n] \mathsf{Pr}[s_t(i) = 1] = \mathsf{Pr}[s_t(i) = -1] = 1/2$.

The measurement vector entries are $\mu_t[i] := h_t(i)s_t(i)$ ($i \in [n]$). Our upper bound only requires limited independence (3-wise for h_t and 5-wise for s_t, respectively). Our lower bounds hold in the stronger model of full independence.

2.2. The median estimator

We say that a key $i \in [n]$ *participates* in bucket $t \in [d]$ when $\mu_t[i] \neq 0$. We denote by $T_i := \{t \mid \mu_t[i] \neq 0\}$ the set of buckets that i participates in. Note that with `CountSketch` we have $|T_i| = d/b$ since i participates in exactly one bucket in each set of b buckets and with `BCountSketch` we have $\mathbb{E}[|T_i|] = d/b$ since i participates in each bucket with probability $1/b$. Also note that with both methods, $\boldsymbol{\mu}_t$ for $t \in T_i$ are i.i.d.

Note that for all $i \in [n]$ it holds that $\mathbb{E}_{\boldsymbol{\rho}}[\mu_t[i] \cdot c_t \mid t \in T_i] = v[i]$. For each key $i \in [n]$ we get a multiset of unbiased independent weak estimates of the value $v[i]$ (one for each $t \in T_i$):

$$V(i) := \{\mu_t[i] \cdot c_t \mid \mu_t[i] \neq 0\}.$$

We use these estimates to determine if i should be reported as a heavy hitter and if so, its reported estimated value. The classic `CountSketch` estimator (15) uses the median of these values: $\hat{v}[i] := \text{median}V(i)$.

2.3. Utility of `CountSketch` and `BCountSketch`

The median estimator guarantees that for $\delta \in (0, 1)$ and $d = O(b \cdot \log(1/\delta)$, $\mathsf{Pr}_{\boldsymbol{\rho}\sim\mathcal{D}}\left[(v[i] - \hat{v}[i])^2 > \frac{1}{b}\|\boldsymbol{v}_{\mathsf{tail}[b]}\|_2^2\right] \le \delta$, where $\boldsymbol{v}_{\mathsf{tail}[k]}$, the k-tail of $\boldsymbol{v}$, is a vector obtained from $\boldsymbol{v}$ by replacing its k largest entries in magnitude with 0. The analysis extends to `BCountSketch` (that has the same distribution of the independent bucket estimates except that their number is d/b in expectation and not exact). The median estimator is unbiased whereas other quantiles of $V(i)$ may not be, but importantly for our robust weight estimation, the stated error bound holds for any *quantile* $\hat{v}[i]$ of $V(i)$ in a range $(1/2 - \phi, 1/2 + \phi)$ of $V(i)$, where the constant $\phi < 1/2$ can be tuned by the constant factors of setting the sketch parameters (38).

The following ℓ_∞/ℓ_2 guarantee is obtained using a union bound over keys (15) (for $\alpha, \delta \in (0, 1/4)$ and sketch parameters $b = O(k/\alpha)$ and $d = O((k/\alpha)\log(n/\delta))$):

$$\mathsf{Pr}_{\boldsymbol{\rho}\sim\mathcal{D}}\left[\|\hat{\boldsymbol{v}}^{(\boldsymbol{\rho})} - \boldsymbol{v}\|_\infty^2 > \frac{\alpha}{k}\|\boldsymbol{v}_{\mathsf{tail}[k/\alpha]}\|_2^2\right] \le \delta, \quad (1)$$

where $\hat{\boldsymbol{v}}^{(\boldsymbol{\rho})}$ is an approximation of $\boldsymbol{v}$ that is computed from $\texttt{Sketch}_{\boldsymbol{\rho}}(\boldsymbol{v})$. For the heavy hitters problem, we return the set of $k(1+1/(1-2\alpha))$ keys with largest estimates, along with their estimated values.

When we have m different *non-adaptive* inputs $(\boldsymbol{v}_q)_{q\in[m]}$, a simple union bound argument with (1) provides that with a sketch parameters $b = O(k/\alpha)$ and $d = O((k/\alpha)\log(nm/\delta))$:

$$\mathsf{Pr}_{\boldsymbol{\rho}\sim\mathcal{D}}\left[\forall q\in[m],\ \|\hat{\boldsymbol{v}}_q^{(\boldsymbol{\rho})}-\boldsymbol{v}_q\|_\infty^2 \le \frac{\alpha}{k}\|(\boldsymbol{v}_q)_{\mathsf{tail}[k]}\|_2^2\right] \ge 1-\delta\ . \tag{2}$$

That is, the number of inputs for which we can guarantee utility with high probability grows *exponentially* with the size of the sketch. As mentioned in the introduction, we shall see that the median estimator is not robust in adaptive settings, where we can only guarantee utility for number of inputs that grows *linearly* with the sketch size, matching a trivial upper bound.

3. Sign-Alignment Estimators

We propose sign-alignment estimators (with `CountSketch` and `BCountSketch`) that determine whether a key i is reported as a potential heavy hitter based on the number of buckets for which the signs of $\mu_t[i]\cdot c_t$ align.

For $\boldsymbol{\mu}\in\mathrm{Supp}(\mathcal{B})$, $\boldsymbol{v}\in\mathbb{R}^n$, and $i\in[n]$, we define the predicates

$$h^+_{\boldsymbol{v},i}(\boldsymbol{\mu}) := \langle\boldsymbol{\mu},\boldsymbol{v}\rangle\mu[i] > 0$$
$$h^-_{\boldsymbol{v},i}(\boldsymbol{\mu}) := \langle\boldsymbol{\mu},\boldsymbol{v}\rangle\mu[i] < 0\ .$$

We show that for a key i that participates in the bucket t, if i is heavy then the sign of $v[i]$ is very likely to agree with the sign of the bucket estimate $\mu[i]c_t$ but when there are many keys that are heavier than i (i lies in the "tail") then such agreement is less likely. For $\boldsymbol{v}$ and $i\in[n]$ we accordingly define the probabilities that these predicates are satisfied by $\boldsymbol{\mu}$, conditioned on i participating, as

$$p^+(\boldsymbol{v},i) := \mathsf{Pr}_{\boldsymbol{\mu}}[h^+_{\boldsymbol{v},i}(\boldsymbol{\mu}) \mid \mu[i]\neq 0] = b\cdot\mathsf{Pr}_{\boldsymbol{\mu}}[h^+_{\boldsymbol{v},i}(\boldsymbol{\mu})]$$
$$p^-(\boldsymbol{v},i) := \mathsf{Pr}_{\boldsymbol{\mu}}[h^+_{\boldsymbol{v},i}(\boldsymbol{\mu}) \mid \mu[i]\neq 0] = b\cdot\mathsf{Pr}_{\boldsymbol{\mu}}[h^-_{\boldsymbol{v},i}(\boldsymbol{\mu})]$$
$$p(\boldsymbol{v},i) := \max\{p^+(\boldsymbol{v},i), p^+(\boldsymbol{v},i)\}$$

The intuition is that when $v[i]^2 \ll \frac{\alpha}{k}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2^2$, we expect $|v(i)| \ll |c_t|$ and therefore $p^+(\boldsymbol{v},i)\approx p^-(\boldsymbol{v},i)\approx 1/2$ and thus $p(\boldsymbol{v},i)\approx 1/2$. When $v[i]^2 > \frac{1}{k}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2^2$ we expect $v[i]\mu_t[i]\approx c_t$ and $\mathsf{Pr}[c_t\cdot\mu_t[i]\cdot\mathrm{sgn}(v[i])>0 \mid t\in T_i]\approx 1$ and hence $p(\boldsymbol{v},i)\approx 1$.

Lemma 3.1. *There are constants C_a and C_b and $1/2<\tau_a<\tau_b<1$ such that for all $\boldsymbol{v}\in\mathbb{R}^n$, for all $i\in[n]$*

- *If $v[i]^2 > \frac{C_b^2}{b}\|\boldsymbol{v}_{\mathsf{tail}[b/C_b^2]}\|_2^2$ then $p^{\mathrm{sgn}(v[i])}(\boldsymbol{v},i)\ge\tau_b$ (and therefore $p(\boldsymbol{v},i)\ge\tau_b$)*
- *If $v[i]^2 \le \frac{1}{b}\|\boldsymbol{v}_{\mathsf{tail}[C_a\cdot b]}\|_2^2$ then $p(\boldsymbol{v},i)\le\tau_a$.*

Corollary 3.2. *Consider sketches with width $b=C_b^2\cdot k$ and define*

$$\textit{heavy}(\boldsymbol{v}) := \{i\in[n]\mid p(\boldsymbol{v},i)\ge\tau_b\} \tag{3}$$
$$\textit{suspect}(\boldsymbol{v}) := \{i\in[n]\mid p(\boldsymbol{v},i)\ge\tau_a\}\ . \tag{4}$$

Then the set heavy($\boldsymbol{v}$) contains all heavy hitter keys of $\boldsymbol{v}$ and $|\textit{suspect}(\boldsymbol{v})|\le(C_a+1)\cdot b=(C_a+1)\cdot C_b^2\cdot k$.

It follows from Corollary 3.2 that a set $\mathtt{K}(\boldsymbol{v})$ that includes all heavy($\boldsymbol{v}$) keys and only suspect($\boldsymbol{v}$) keys is a correct solution of the heavy hitters problem. Our sign-alignment estimators are specified by two components. The first component is obtaining estimates $\hat{p}^\sigma(\boldsymbol{v},i)$ of $p^\sigma(\boldsymbol{v},i)$ given a query $\texttt{Sketch}_\rho(\boldsymbol{v})$ with $b=C_b^2k$, a key $i\in[n]$, and $\sigma\in\{-1,+1\}$. We shall see (Section 3.3) that simple averaging suffices for the oblivious setting but more nuanced methods (Section 4) are needed for robustness. The second component is the estimator that uses these estimates to compute an output set $\mathtt{K}(\boldsymbol{v})$. We present two methods, *threshold* (Section 3.1) for arbitrary queries and *stable* (Section 3.2) for continuous reporting.

Remark 3.3. *Sign-alignment estimators only report high-alignment keys $i\in[n]$ that "dominate" most of their buckets. This because with probability $2p(\boldsymbol{v},i)-1$, the magnitude of contribution of keys $[n]\setminus\{i\}$ to a bucket is smaller than $|v[i]|$. In particular, vectors $\boldsymbol{v}$ with no heavy keys (empty suspect($\boldsymbol{v}$)) will have no reported keys. This is a distinct advantage over estimators that simply report $O(k)$ keys with highest estimates.*

3.1. The Threshold Estimator

A threshold sign-alignment estimator output the set of keys:

$$\mathtt{K}(\boldsymbol{v}) = \{i\in[n]\mid \max\{\hat{p}^+(\boldsymbol{v},i),\hat{p}^-(\boldsymbol{v},i)\}\ge\tau_m\}, \tag{5}$$

where $\tau_m := (\tau_a+\tau_b)/2$.

Lemma 3.4. *(correctness of threshold estimators) If for each query vector $\boldsymbol{v}$, $i\in[n]$, and $\sigma\in\{+,-\}$ the estimates $\hat{p}^\sigma(\boldsymbol{v},i)$ satisfy*

$$|\hat{p}^\sigma(\boldsymbol{v},i)-p^\sigma(\boldsymbol{v},i)| \le \frac{\tau_b-\tau_a}{2}, \tag{6}$$

then the output $\mathtt{K}(\boldsymbol{v})$ is correct.

Proof.

$$i\notin\mathtt{K}(\boldsymbol{v}) \implies \hat{p}^\sigma(\boldsymbol{v},i)<\tau_m \implies p^\sigma(\boldsymbol{v},i)\le\hat{p}^\sigma(\boldsymbol{v},i)+$$
$$|\hat{p}^\sigma(\boldsymbol{v},i)-p^\sigma(\boldsymbol{v},i)| < \tau_m+\frac{1}{2}(\tau_b-\tau_a)=\tau_b \implies i\notin\mathtt{heavy}(\boldsymbol{v})$$
$$i\in\mathtt{K}(\boldsymbol{v}) \implies \hat{p}^\sigma(\boldsymbol{v},i)\ge\tau_m \implies p^\sigma(\boldsymbol{v},i)\ge\hat{p}^\sigma(\boldsymbol{v},i)-$$
$$|\hat{p}^\sigma(\boldsymbol{v},i)-p^\sigma(\boldsymbol{v},i)| \ge \tau_m-\frac{1}{2}(\tau_b-\tau_a)=\tau_a \implies i\in\mathtt{suspect}(\boldsymbol{v})$$

□

3.2. The Stable Estimator

This estimator is designed for a *continuous reporting* version of the heavy hitters problem and is beneficial when the input sequence is of related vectors (as in streaming). In this case we report K continuously and modify it as needed due to input changes. In these applications we desire *stability* of K, in the sense of avoiding thrashing, where a borderline key exits and re-enters K following minor updates. We shall see that stability can significantly improve robustness guarantees, as we only need to account for *changes* in the reported set instead of the total size of each reported set. Our stable estimator uses two threshold values:

$$\tau_{m_1} := \tau_a + \frac{1}{5}(\tau_b - \tau_a) \tag{7}$$

$$\tau_{m_2} := \tau_b - \frac{1}{5}(\tau_b - \tau_a)\ . \tag{8}$$

A key $i \notin$ K enters the reported set when $\max\{\hat{p}^+(\boldsymbol{v}, i), \hat{p}^-(\boldsymbol{v}, i)\} \geq \tau_{m_2}$. A key $i \in$ K exits the reported set when $\max\{\hat{p}^+(\boldsymbol{v}, i), \hat{p}^-(\boldsymbol{v}, i)\} < \tau_{m_1}$.

Lemma 3.5. *(correctness of stable estimators) If for all queries $\boldsymbol{v}$, keys $i \in [n]$, and $\sigma \in \{-1, +1\}$, our estimates satisfy*

$$|\hat{p}^\sigma(\boldsymbol{v}, i) - p^\sigma(\boldsymbol{v}, i)| \leq \frac{1}{5}(\tau_b - \tau_a) \tag{9}$$

then the output of the stable estimator is correct (K includes all heavy($\boldsymbol{v}$) keys and only suspect($\boldsymbol{v}$) keys). Moreover, the reporting status of a key can change only when $p(\boldsymbol{v}, i)$ changes by at least $\frac{1}{5}(\tau_b - \tau_a)$.

Proof. Similar to that of Lemma 3.4. □

3.3. Basic estimates

The *basic* estimates are simply averages over buckets:

$$\hat{p}^\sigma(\boldsymbol{v}, i) := \frac{b}{d} \sum_{t \in [d]} \mathbb{1}\{\mu_t[i] \cdot c_t \cdot \sigma > 0\}\ . \tag{10}$$

Lemma 3.6. *When $\boldsymbol{v}$ is chosen non-adaptively, then for any constant τ_Δ and $\beta > 0$, using $d = O(\frac{1}{\tau_\Delta^2} b \log(mn/\beta)$,*

$$\mathsf{Pr}[\max_{q \in [m], i \in [n], \sigma \in \{-1,+1\}} |p^\sigma(\boldsymbol{v}, i) - \hat{p}^\sigma(\boldsymbol{v}, i)| > \tau_\Delta] \leq \beta$$

Proof. From multiplicative Chernoff bound, we obtain that

$$\mathsf{Pr}[|p^\sigma(\boldsymbol{v}, i) - \hat{p}^\sigma(\boldsymbol{v}, i)| > \tau_\Delta] \leq 2e^{-\frac{1}{3}\tau_\Delta^2 \frac{d}{b}}\ .$$

We then apply a union bound over the $2mn$ queries. □

It follows that in the non-adaptive setting the sign-alignment estimators are correct with $d = O(b \log(mn/\beta)$. This matches the utility guarantees provided with the median estimator (Section 2.3). We shall see however that as is the case for the median estimator, our sign-alignment estimators with the basic estimates are non-robust in adaptive settings. The robust estimators we introduce in Section 4 use estimates $\hat{p}^\sigma(\boldsymbol{v}, i)$ that are more nuanced.

4. Robust Estimators

We present two sign-alignment estimators for BCountSketch that are robust against adaptive adversaries: A robust version of the threshold estimator of Section 3.1, that is described as Algorithm 2 in Section 4.1 (correctness proof provided in Section A), and a robust version of the stable estimator of Section 3.2 that is described as Algorithm 4 in Appendix D.

In the introduction we stated robustness guarantees in terms of the size of the query sequence $Q = (\boldsymbol{v}_q)_{q=1}^m$: A sketch with parameters (d, b) provides guarantees when $|Q| = \tilde{O}((d/b)^2)$. But this accounting is coarse as it incurs the same "cost" per query $\boldsymbol{v}_q$ even when very few keys are actually reported or when there is little or no change between reported sets on consecutive inputs. We introduce a refined property of sequences, its λ-number (λ_Q), that can be much smaller than $|\lambda_Q|$ and captures smaller output sizes (number of keys reported) with the threshold estimator and *changes* in the reported set with the stable estimators. We then establish robustness guarantees in terms of $\lambda_Q = \tilde{O}((d/b)^2)$.

The λ-number we use to analyze our robust threshold estimator (Algorithm 2) accounts only for potential reporting of each key, namely, the number of times it occurs in $\mathrm{suspect}(\boldsymbol{v}_q)$. From Remark 3.3, the gain can be considerable on inputs $\boldsymbol{v}_q$ with a small number of heavy keys or with no heavy keys.

Definition 4.1. *(λ-number of an input sequence) For an input sequence $Q = (\boldsymbol{v}_q)_{q=1}^m$ and a key i, we define*

$$\lambda_{Q,i} := \sum_{\boldsymbol{v} \in Q} \mathbb{1}\{i \in \mathrm{suspect}(\boldsymbol{v})\} \tag{11}$$

to be the number of vectors $\boldsymbol{v} \in Q$ for which i is in $\mathrm{suspect}(\boldsymbol{v})$. *For an input sequence Q, define*

$$\lambda_Q := \min\left\{\max_i \lambda_{Q,i}, \frac{1}{C_a \cdot b} \sum_i \lambda_{Q,i}\right\}\ . \tag{12}$$

The λ-number we use to analyze our stable robust estimator (Algorithm 4) accounts only for *changes* in the output between consecutive inputs. This is particularly beneficial for streaming, where updates to the input are incremental and hence consecutive outputs tend to be similar. Our redefined $\lambda_{Q,i}$ bounds the number of times that the key i may enter or exit the reported set when a stable estimator is used. We then redefine λ_Q accordingly as in (12). Note that the redefined values always satisfy $\lambda_Q \leq 2|Q|$ (as $\lambda_{Q,i}$ is at most twice (11)) but it is possible to have $\lambda_Q \ll |Q|$, allowing for robustness on longer streams with the same budget. The approach of accounting for changes in the output in the context of robust streaming was first proposed in (12) and we extend their use of the term *flip number*.

Definition 4.2. *(flip number of a key) Consider an input sequence Q. We say that a key i is high at step q if $p(\boldsymbol{v}_q, i) \geq \tau_b - \frac{2}{5}(\tau_b - \tau_a)$ and is low at step q if $p(\boldsymbol{v}_q, i) \leq \tau_a + \frac{2}{5}(\tau_b - \tau_a)$. The* flip number *of a key $\lambda_{Q,i}$ is defined as the number of transitions from low to high or vice versa (skipping steps where it is neither).*

Remark 4.3. *Consider the stable estimator when $|\hat{p}^\sigma_{\boldsymbol{v}_q,i} - p^\sigma_{\boldsymbol{v}_q,i}| \leq \frac{1}{5}(\tau_b - \tau_a)$ for all σ, q, i. The number of times key i enters or exits the reported set is at most $\lambda_{Q,i}$.*

Our robust estimators provide the following guarantees:

Theorem 4.1. *Our robust threshold (Algorithm 2) and stable (Algorithm 4) estimators, with appropriate setting of the constants, provide the following guarantees (each for its respective definition of λ_Q): Let C_3, c_1 be appropriate constants. Let $\beta > 0$. Consider an execution of our robust estimator with adaptive inputs $Q = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{|Q|})$, access limit L, and i.i.d initialization of $(\boldsymbol{\mu}_t)_{t\in[d]}$ and*

$$d \geq C_3 \cdot b \cdot \sqrt{L} \log(L \log(m \cdot n \cdot b \cdot L)) \log(m \cdot n \cdot L) \log(\frac{m \cdot n}{\beta}) \tag{13}$$

Then if $\lambda_Q \leq c_1 L$ and $|Q| \leq m$, with probability $1 - \beta$ all outputs are correct.

Restated, we obtain that a sketch with parameters (n, d, b), with our robust estimators, provides robustness to adaptive inputs Q with $\lambda_q = \widetilde{O}((d/b)^2)$.

4.1. The Robust Threshold Estimator

Our robust threshold estimator is provided as Algorithm 2 (The constants C_1, C_2 will be chosen sufficiently large). The algorithm initializes a `ThresholdMonitor` structure TM (34) (see Algorithm 1) over the dataset of the d measurement vectors (buckets). A `ThresholdMonitor` inputs a predicate that is defined over $\text{Supp}(\mathcal{B})$ and a threshold value and tests whether a noisy count of the predicate over $(\boldsymbol{\mu}_t)_{t\in[d]}$ exceeds the threshold. It has the property that the privacy budget is only charged on queries where the noisy count exceeds the threshold and only buckets on which the predicate evaluates as correct are charged. Note that it is not necessary for the algorithm to track λ_Q in order to determine when to halt (instead, it can halt when too many buckets of the same key turn inactive.) The access limit L specifies how many times we can charge a bucket before it inactivates. The values $\lambda_{Q,i}$ constitute upper bounds on the number of times the buckets of key i get charged during the execution. Additional details on TM and the correctness proof of our algorithm are provided in Appendix A.

For each query vector, the estimator loops over all keys $i \in [n]$ and tests whether the count of the predicates $h^\sigma_{\boldsymbol{v},i}$ over active buckets, with noise added, exceeds a threshold. If so, the key is reported and the access count for the buckets that contributed to the count is incremented. Otherwise, the key is not reported. In Appendix C we show that we can make this more efficient by using a non-robust heavy-hitters sketch to exclude testing of keys that are highly unlikely to be reported. The robust estimator as presented only reports a set of keys `K`. In Appendix B we describe how weight estimates can be reported as well for $i \in$ `K`, by only doubling the "robustness budget" (The TM access limit).

Algorithm 1: `ThresholdMonitor` (34)

```
Input: Database $S \in X^*$, privacy parameters $\varepsilon, \delta$, access limit $L$, adaptive
       sequence of queries $(f_i, s_i, \tau_i)_i$ for predicates $f_i : X \to \{0,1\}$,
       $s_i \in \{+1, -1\}$, and $\tau_i \in \mathbb{N}$
foreach $x \in S$ do $c(x) \leftarrow 0$ // Initialize counters
$\Delta \leftarrow \frac{1}{\varepsilon}\log\left(\frac{1}{\delta}\right)\log\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)$
foreach query $(f_i, \sigma_i, \tau_i)$ do // round $i$
    Choose $a \sim \text{Lap}(10\Delta)$ and $b \sim \text{Lap}(\frac{1}{\varepsilon}\log\frac{1}{\delta})$
    $\hat{f}_i \leftarrow$
      $f_i(S) + a + \mathbb{1}_{\{s_i=1\}} \cdot \min\{\Delta, b\} + \mathbb{1}_{\{s_i=-1\}} \cdot \max\{-\Delta, b\}$
    if $\hat{f}_i \cdot s_i < \tau_i \cdot s_i$ then
        Output $\perp$.
    else
        foreach $x \in S$ do $c(x) \leftarrow c(x) + f_i(x)$
        Delete from $S$ every element $x$ such that $c(x) \geq L$
        Output $\top$.
```

Algorithm 2: Robust Threshold `BCountSketch` Estimator

```
Input: Sketch parameters $(n, d, b)$, Access limit $L$, upper bound $m \geq L$ on
       the number of queries.
$(\boldsymbol{\mu}_t)_{t\in[d]} \leftarrow$ initialized BCountSketch with parameters $(n, d, b)$
$\tau_m = (\tau_a + \tau_b)/2$ // threshold
initialize ThresholdMonitor TM:
    TM.$S \leftarrow (\boldsymbol{\mu}_t)_{t\in[d]}$ // Measurement vectors
    TM.$(\varepsilon, \delta) \leftarrow \left(\frac{C_1}{\sqrt{L}}, \frac{C_2}{n \cdot m \cdot b \cdot L}\right)$ // privacy parameters
    TM.$L \leftarrow L$ // access limit
foreach query sketch $(c_t = \langle \boldsymbol{\mu}_t, \boldsymbol{v} \rangle)_{t\in[d]}$ of a vector $\boldsymbol{v} \in \mathbb{R}^n$ do
    K $\leftarrow \emptyset$        // Initialize list of output keys
    foreach $i \in [n]$ do // loop over keys
        Define $f^+ : [d] \to \{0,1\}$, where $f^+(t) = \mathbb{1}_{\mu_t[i] \cdot c_t > 0}$
        Define $f^- : [d] \to \{0,1\}$, where $f^-(t) = \mathbb{1}_{\mu_t[i] \cdot c_t < 0}$
        if TM.$query(f^+, +1, \frac{d}{b}\tau_m) = \top$ or
          TM.$query(f^-, +1, \frac{d}{b}\tau_m) = \top$ then
            K $\leftarrow$ K $\cup \{i\}$
    return K
```

Acknowledgements Edith Cohen is partially supported by Israel Science Foundation (grant no. 1595/19). Moshe Shechner is partially supported by Israel Science Foundation (grants no. 1595/19 and 1871/19). Uri Stemmer is partially supported by the Israel Science Foundation (grant 1871/19) and by Len Blavatnik and the Blavatnik Family foundation.

References

[1] Amirali Aghazadeh, Ryan Spring, Daniel LeJeune, Gautam Dasarathy, Anshumali Shrivastava, and Richard G. Baraniuk. MISSION: ultra large-scale feature selection using count-sketches. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 80–88. PMLR, 2018.

[2] Thomas D. Ahle, Michael Kapralov, Jakob Bæk Tejs Knudsen, Rasmus Pagh, Ameya Velingker, David P. Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*. SIAM, 2020.

[3] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 459–467.

[4] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58:137–147, 1999.

[5] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[6] Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. A framework for adversarial streaming via differential privacy and difference estimators. *CoRR*, abs/2107.14527, 2021.

[7] Raef Bassily and Yoav Freund. Typical stability, 2016.

[8] Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan R. Ullman. Algorithmic stability for adaptive data analysis. *SIAM J. Comput.*, 50(3), 2021.

[9] Amos Beimel, Haim Kaplan, Yishay Mansour, Kobbi Nissim, Thatchaphol Saranurak, and Uri Stemmer. Dynamic algorithms against an adaptive adversary: Generic constructions and lower bounds. *CoRR*, abs/2111.03980, 2021.

[10] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. *Theory Comput.*, 12(1):1–61, 2016.

[11] Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. Adversarially robust streaming via dense-sparse tradeoffs. *CoRR*, abs/2109.03785, 2021.

[12] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *SIGMOD Rec.*, 50(1):6–13, 2021.

[13] Mark Bun, Cynthia Dwork, Guy N. Rothblum, and Thomas Steinke. Composable and versatile privacy via truncated CDP. In *STOC*, pages 74–86. ACM, 2018.

[14] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649. IEEE Computer Society, 2015.

[15] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, ICALP '02, page 693–703. Springer-Verlag, 2002.

[16] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2285–2294. JMLR.org, 2015.

[17] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing Convolutional Neural Networks in the Frequency Domain. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, pages 1475–1484. ACM, 2016.

[18] Edith Cohen, Rasmus Pagh, and David Woodruff. Wor and p's: Sketches for ℓ_p-sampling without replacement. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21092–21104. Curran Associates, Inc., 2020.

[19] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *STOC*, pages 117–126. ACM, 2015.

[20] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[21] Vitaly Feldman and Thomas Steinke. Generalization for adaptively-chosen estimators via stable median. In *COLT*, volume 65 of *Proceedings of Machine Learning Research*, pages 728–757. PMLR, 2017.

[22] David A. Freedman. A note on screening regression equations. *The American Statistician*, 37(2):152–155, 1983.

[23] Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Minimum cut in o(m $\log^2$ n) time. In *ICALP*, volume 168 of *LIPIcs*, pages 57:1–57:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[24] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[25] Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. *arXiv preprint arXiv:2106.04378*, 2021.

[26] Maximilian Gutenberg and Christian Wulff-Nilsen. *Deterministic Algorithms for Decremental Approximate Shortest Paths: Faster and Simpler*, pages 2522–2541. 01 2020.

[27] Maximilian Probst Gutenberg and Christian Wulff-Nilsen. Decremental sssp in weighted digraphs: Faster and against an adaptive adversary. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, page 2542–2561, USA, 2020. Society for Industrial and Applied Mathematics.

[28] M. Hardt and J. Ullman. Preventing false discovery in interactive data analysis is hard. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 454–463. IEEE Computer Society, 2014.

[29] Moritz Hardt and David P. Woodruff. How robust are linear sketches to adaptive inputs? In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 121–130, New York, NY, USA, 2013. Association for Computing Machinery.

[30] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. In *Annual Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[31] John P. A. Ioannidis. Why most published research findings are false. *PLoS Med*, (2):8, 2005.

[32] Akshay Kamath, Eric Price, and David P. Woodruff. *A Simple Proof of a New Set Disjointness with Applications to Data Streams*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, DEU, 2021.

[33] Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. In *COLT*, volume 125 of *Proceedings of Machine Learning Research*, pages 2263–2285. PMLR, 2020.

[34] Haim Kaplan, Yishay Mansour, and Uri Stemmer. The sparse vector technique, revisited. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 2747–2776. PMLR, 2021.

[35] Aryeh Kontorovich, Menachem Sadigurschi, and Uri Stemmer. Adaptive data analysis with correlated observations, 2022.

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

[37] Paul M. Lukacs, Kenneth P. Burnham, and David R. Anderson. Model selection bias and Freedman's paradox. *Annals of the Institute of Statistical Mathematics*, 62(1):117, 2009.

[38] Gregory T. Minton and Eric Price. *Improved Concentration Bounds for Count-Sketch*, pages 669–686.

[39] Ilya Mironov, Moni Naor, and Gil Segev. Sketching in adversarial environments. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 651–660, New York, NY, USA, 2008. Association for Computing Machinery.

[40] J. Misra and D. Gries. Finding repeated elements. Technical report, Cornell University, 1982.

[41] John E. Moody and Christian J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Comput.*, 1(2):281–294, 1989.

[42] Jelani Nelson, Huy L. Nguyễn, and David P. Woodruff. On deterministic sketching and streaming for sparse recovery and norm estimation. *Lin. Alg. Appl.*, 441:152–167, January 2014. Preliminary version in RANDOM 2012.

[43] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[44] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. FetchSGD: Communication-efficient federated learning with sketching. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8253–8265. PMLR, 13–18 Jul 2020.

[45] Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, Alex Strehl, and S. V. N. Vishwanathan. Hash kernels. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 496–503, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[46] Yossi Shiloach and Shimon Even. An on-line edge-deletion problem. *J. ACM*, 28(1):1–4, jan 1981.

[47] Ryan Spring, Anastasios Kyrillidis, Vijai Mohan, and Anshumali Shrivastava. Compressing gradient optimizers via count-sketches. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5946–5955. PMLR, 09–15 Jun 2019.

[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[49] David Wajc. *Rounding Dynamic Matchings against an Adaptive Adversary*. Association for Computing Machinery, New York, NY, USA, 2020.

[50] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1113–1120, New York, NY, USA, 2009. Association for Computing Machinery.

[51] David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2021.

A. Proofs for the robust threshold estimator

In this Section we provide a proof of correctness of our robust threshold estimator. We use $\ell = d/b$ and the constants C_a, C_b, τ_a, τ_b as in Lemma 3.1. Let $\tau_m := (\tau_a + \tau_b)/2$ and $\tau_\Delta := (\tau_b - \tau_a)/10$. We establish the following.

Theorem A.1. *Let C_1, C_2, C_3, c_1 be appropriate constants. Let $\beta > 0$. Consider an execution of Algorithm 2 with adaptive inputs $Q = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{|Q|})$ and i.i.d initialization of $(\boldsymbol{\mu}_t)_{t\in[d]}$. Set*

$$d \geq C_3 \cdot b \cdot \sqrt{L}\log(L\log(m\cdot n\cdot b\cdot L))\log(m\cdot n\cdot L)\log(\frac{m\cdot n}{\beta}) \tag{14}$$

Then if $\lambda_Q \leq c_1 L$, with probability $1-\beta$ all outputs are correct.

A.1. Tools from Differential Privacy

First, we need to introduce necessary tools from Differential Privacy.

Theorem A.2. *(Generalization property of DP (19; 8; 21)) Let $\mathcal{A} : X^d \to 2^X$ be an (ξ, η)-differentially private algorithm that operates on a database of size d and outputs a predicate $h : X \to \{0, 1\}$. Let $\mathcal{D}$ be a distribution over X, let S be a database containing d i.i.d. elements from $\mathcal{D}$, and let $h \leftarrow \mathcal{A}(S)$. Then for any $T \geq 1$ we have that*

$$\Pr_{\substack{S\sim\mathcal{D},\\ h\leftarrow\mathcal{A}(S)}}\left[e^{-2\xi}h(\mathcal{D}) - \frac{1}{d}\sum_{X\in\mathcal{S}}h(X) > \frac{4}{\xi d}\log(T+1) + 2T\eta\right] < \frac{1}{T}.$$

THE FINE-GRAINED SPARSE VECTOR TECHNIQUE

We will employ the (fine-grained) sparse vector of (34) described in Algorithm 1 (`ThresholdMonitor`). Algorithm `ThresholdMonitor` has the following utility and privacy guarantees:

Theorem A.3 (Utility guarantee (34)). *Consider an execution of Algorithm `ThresholdMonitor` on a database S and on a sequence $(f_i, s_i, \tau_i)_{i\in[r]}$ of adaptively chosen queries. Let S_i denote the database S as it is before answering the i-th query. With probability at least $1-\beta$, it holds that for all $i \in [r]$*

- *If the output is $\top$ then $f_i(S_i)\cdot s_i \geq \tau_i \cdot s_i - O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\log(\frac{1}{\varepsilon}\log\frac{1}{\delta})\log\frac{r}{\beta}\right)$.*
- *If the output is $\perp$ then $f_i(S_i)\cdot s_i \leq \tau_i \cdot s_i + O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\log(\frac{1}{\varepsilon}\log\frac{1}{\delta})\log\frac{r}{\beta}\right)$.*

Theorem A.4 (Privacy guarantee (34)). *Algorithm `ThresholdMonitor` is $(O(\sqrt{L}\varepsilon + L\varepsilon^2), O(L\delta))$-differentially private.*

A.2. Proof overview of Theorem A.1

Algorithm 2 issues to TM $2n$ count queries for each input vector $\boldsymbol{v}_q$. These queries correspond to predicates h of the form $h^\sigma_{\boldsymbol{v}_q,i}$ (sign $\sigma \in \{+,-\}$, key $i \in [n]$ and query $\boldsymbol{v}_q$).

For each predicate h, the respective count of the TM is computed over buckets that are active at the time of query:

$$\begin{aligned} F &:= \sum_{t\in[d]} \mathbb{1}_{(\mu_t[i]\cdot c_t\cdot\sigma>0)\wedge t \text{ is active}} \\ &= d\cdot \Pr_{t\sim[d]}[h(\boldsymbol{\mu}_t)\wedge t \text{ is active}]\ . \end{aligned}$$

The TM adds noise to F to obtain $\hat{F}$. The TM outputs $\top$ for the query h if and only if $\hat{F} \geq \frac{d}{b}\tau_m$. A key i is reported if and only if the TM output is $\top$ for at least one of its two queries. Equivalently, the inclusion of each key i in the output set $\mathtt{K}(\boldsymbol{v}_q)$ is determined as in (5) using the respective approximate values $\hat{p}^\sigma(\boldsymbol{v}_q, i) := \frac{b}{d}\hat{F}$.

It follows from Lemma 3.4 that if all our noisy counts $\hat{F}$ over buckets are within $\frac{d}{b}(\tau_b - \tau_a)/2$ of their expectation over $\mathcal{B}$ then the output is correct. We will show that this happens with probability $1-\beta$.

We will bound the "error" of $\hat{F}$ by separately bounding the contributions of different sources of error. We show that with probability $1-\beta$ the additive error is at most $\frac{5}{b}\tau_\Delta$ for all these estimates.

- One source of error is due to TM not counting inactive buckets (those that reached the access limit L by TM). We introduce the notion of "useful" buckets (see Section A.3), where usefulness is a deterministic property of the input sequence and has the property that all useful buckets remain active. We show that *in expectation*, for each key i, a $(1-\tau_\Delta)$ fraction of the buckets that a key i participates in are useful. Hence in expectation the contribution to the error is bounded by τ_Δ/b.

- Another source of error is due to the noise added by TM. We use the parameter settings and Theorem A.3 to bound the maximum error over all queries by τ_Δ/b with probability $1-\beta/2$.

- We establish correctness under the following assumption (see Section A.4). We treat the query vectors $Q=(\boldsymbol{v}_q)$ as fixed and assume the buckets $(\boldsymbol{\mu}_t)$ satisfy the following: We formulate a set H of $O(mn)$ predicates over $\mathcal{B}$ that depend on the query vectors $(\boldsymbol{v}_q)$ so that all have expectation $\leq 1/b$ and the expected value of all these predicates is approximated by the sample $(\boldsymbol{\mu}_t)_{t\in[d]}$ to within an additive error of τ_Δ/b. The set H includes all predicates $h^\sigma_{\boldsymbol{v}_q,i}$ ($\sigma\in\{-1,1\}, i\in[n], q\in[|Q|]$) and also includes the usefulness predicates over buckets each key i participates in. With this assumption, the total error due to inactive buckets is bounded by $2\tau_\Delta/b$ (combining their expectation and the error). Using the assumption, the error due to estimation of $h^\sigma_{\boldsymbol{v}_q,i}$ is at most τ_Δ/b. Recall also that the error due to noise is τ_Δ/b. Combining, we get an error of $4\tau_\Delta/b$ when the assumption holds.

- We remove the assumption by relating (see Section A.5) the count of our predicates over buckets to its respective expectation over $\mathcal{B}$ using the generalization property of DP (Theorem A.2). The property establishes that even though our query vectors and hence predicates are generated in a dependent way on the sampled buckets, because they are generated in a way that preserves the privacy of the buckets, their average over the sampled buckets still approximates well their expectation. This holds with probability $1-\beta/2$ for all predicates.

A.3. Useful buckets

Definition A.1. *(useful buckets) We say that a bucket with measurement vector $\boldsymbol{\mu}$ is* useful *with respect to Q and access limit L if the total count, over vectors in Q, of* $\mathrm{suspect}(\boldsymbol{v})$ *keys that participate in the bucket is at most L:*

$$\textit{useful}_{Q,L}(\boldsymbol{\mu}) := \sum_{i\in[n]} \mathbb{1}_{\mu[i]\neq 0}\cdot\lambda_{Q,i} \leq L\ .$$

The predicate useful depends on the set Q and applies to all $\boldsymbol{\mu}\in\mathrm{Supp}(\mathcal{B})$ whereas being active applies only to the d buckets $(\boldsymbol{\mu}_t)$ and buckets may become inactive over time. We can relate useful and active buckets as follows:

Remark A.2. *Consider an execution of Algorithm 2 with input sequence Q where* TM *only reports keys from* $\mathrm{suspect}(\boldsymbol{v}_q)$ *for each query $\boldsymbol{v}_q\in Q$. Then all useful buckets $t\in[d]$ remain active throughout the execution. Therefore, for any predicate $h:\mathrm{Supp}(\mathcal{B})$, at any point during the execution, it holds that:*

$$\textsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)\wedge \mathrm{useful}_{Q,L}(\boldsymbol{\mu}_t)] \leq \textsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)\wedge t \textit{ is active}] \leq \textsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)]\ .$$

(for notation convenience we use $\textsf{Pr}_{t\in[d]}h(\boldsymbol{\mu}_t)=\frac{1}{d}\sum_{t\in[d]}\mathbb{1}_{h(\boldsymbol{\mu}_t)}$.)

We next characterize the set of input sequences Q with which most buckets of each key remain useful.

Definition A.3. *(Sequences with enough useful buckets) For $L\in\mathbb{N}$, let $\mathcal{Q}(L)$ be the set of all query sequences $Q=(\boldsymbol{v}_q)_{q\in[m]}$ that satisfy*

$$\forall i\in[n],\ \textsf{Pr}_{\boldsymbol{\mu}}[\neg\mathrm{useful}_{Q,L}(\boldsymbol{\mu})\wedge\mu[i]\neq 0]\leq\frac{\tau_\Delta}{b}\ .$$

Recall that the probability that a key participates in a bucket is $\textsf{Pr}_{\boldsymbol{\mu}}[\mu[i]\neq 0]=1/b$. Therefore the requirement is that for all keys i, a random bucket $\boldsymbol{\mu}$ in which i participates is useful with probability $\geq(1-\tau_\Delta)$.

We state sufficient conditions for $Q\in\mathcal{Q}(L)$:

Lemma A.4. *There is a sufficiently small constant c_1 so that for $Q=(\boldsymbol{v}_q)_{q=1}^m$, $\lambda_Q\leq c_1\cdot L \implies Q\in\mathcal{Q}(L)$. Note that in particular this means that $|Q|\leq c_1\cdot L \implies Q\in\mathcal{Q}(L)$.*

Proof. Fixing a sequence Q with $\lambda_Q \leq c_1 L$, for every $i \in [n]$, we argue as follows. Since $\mathsf{Pr}_{\boldsymbol{\mu}}[\mu[i] \neq 0] = 1/b$, it is sufficient to show that

$$\mathsf{Pr}_{\boldsymbol{\mu}}[\neg\mathrm{useful}_Q(\boldsymbol{\mu}) \mid \mu[i] \neq 0] \leq \tau_\Delta.$$

Consider randomly drawing a bucket $\boldsymbol{\mu}$. Define a random variable

$$X(\boldsymbol{\mu}) = \sum_{q \in [m]} |\{i \in [n] \mid \mu[i] \neq 0 \wedge i \in \mathrm{suspect}(\boldsymbol{v}_q)\}|.$$

For every $j \in [n] \setminus \{i\}$, we have

$$\mathsf{Pr}_{\boldsymbol{\mu}}[\mu[j] \neq 0 \mid \mu[i] \neq 0] = \frac{1}{b}.$$

Then, by Condition (i), (ii) and linearity of expectation, we have

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{\mu}}[X(\boldsymbol{\mu}) \mid \mu[i] \neq 0] &\leq \lambda_{Q,i} + \frac{1}{b} \sum_{j \in [n] \setminus \{i\}} \lambda_{Q,j} \\ &\leq c_1 L + \frac{1}{C_a} c_1 L \qquad\qquad \leq 2c_1 L.\end{aligned}$$

Finally, by Markov's inequality, we have

$$\mathsf{Pr}_{\boldsymbol{\mu}}[\neg\mathrm{useful}_Q(\boldsymbol{\mu}) \mid \mu[i] \neq 0] \leq \mathsf{Pr}_{\boldsymbol{\mu}}[X(\mu) > L \mid \mu[i] \neq 0] \leq 2c_1.$$

Hence, choosing $c_1 < \frac{\tau_\Delta}{2}$ suffices.[4] □

A.4. Correctness with assumption

Our choice of $\mathrm{TM}.(\varepsilon, \delta)$ and the sketch size d allow us to provide a high $(1-\beta)$ probability bound on the maximum amount of noise in all $r = 2 \cdot m \cdot n$ queries issued to the TM:

Remark A.5. *(utility) There are values for C_1, C_2 in the algorithm and C_3 in* (13) *so that the utility guarantee of Theorem A.3 provides*

$$O\left(\frac{1}{\varepsilon} \log \frac{1}{\delta} \log(\frac{1}{\varepsilon} \log \frac{1}{\delta}) \log \frac{2mn}{\beta}\right) \leq \tau_\Delta \frac{d}{b}\ . \tag{15}$$

We establish correctness under the simplifying assumption that sampled (sketch) buckets provide good estimates for the expected counts of our predicates.

Lemma A.6. *Consider an execution as follows of Algorithm 2 where constants are set to provide the utility guarantee* (15)*. Fix the buckets $(\boldsymbol{\mu}_t)_{t \in [d]}$. Set* $\mathrm{TM}.L$ *and fix the input sequence $Q = (\boldsymbol{v}_q)_{q \in [m]} \in \mathcal{Q}(L)$.*

Consider the following set of $n(4m+1)$ predicates $h : \mathrm{Supp}(\mathcal{B}) \to \{0,1\}$:

$$H = \{h^{\pm}_{\boldsymbol{v}_q,i}\}_{q \in [m], i \in [n]} \cup \{h^{\pm}_{\boldsymbol{v}_q,i} \wedge \mathrm{useful}_Q\}_{q \in [m], i \in [n]} \cup \{\neg\mathrm{useful}_Q(\boldsymbol{\mu}) \wedge \mu[i] \neq 0\}_{i \in [n]}\ .$$

If for all $h \in H$, it holds that the average of $\mathbb{1}_h$ over the buckets $(\boldsymbol{\mu}_t)_{t \in [d]}$ approximate well its expectation over $\boldsymbol{\mu} \sim \mathcal{B}$:

$$\left|\mathsf{Pr}_{t \sim [d]} h(\boldsymbol{\mu_t}) - \mathsf{Pr}_{\boldsymbol{\mu} \sim B} h(\boldsymbol{\mu})\right| \leq \frac{1}{b} \tau_\Delta\ . \tag{16}$$

Then Algorithm 2 is correct with probability $1-\beta$, that is,

$$\forall q \in [m],\ \mathrm{heavy}(\boldsymbol{v}_q) \subseteq K_q \subseteq \mathrm{suspect}(\boldsymbol{v}_q)\ .$$

Proof. Algorithm 2 issues to TM $2n$ count queries for each input vector $\boldsymbol{v}_q$. These queries correspond to predicates h of the form $h^{\sigma}_{\boldsymbol{v}_q,i}$ (sign $\sigma \in \{+,-\}$, key $i \in [n]$ and query $\boldsymbol{v}_q$).

[4] The constants can be improved using Chebyshev inequality and pairwise independence of two keys mapping into the same bucket.

For each predicate h, the respective count of the TM is computed over buckets that are active at the time of query:

$$F := d \cdot \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) \wedge t \text{ is active}] \;.$$

(recall that the set of active buckets may decrease over time). The TM adds noise to F to obtain $\hat{F}$. It then outputs $\top$ for the query h if and only if $\hat{F} \geq \frac{d}{b}\tau_m$.

From Remark A.5, with probability $1-\beta/2$, TM satisfies the utility guarantee for all $2mn$ queries, that is

$$|\hat{F} - F| \leq \frac{d}{b}\tau_\Delta \;. \tag{17}$$

We next bound the maximum over keys i of the number of buckets amongst $(\boldsymbol{\mu}_t)_{t\in[d]}$ that i participates in and are not useful.

$$\begin{aligned}
&\mathsf{Pr}_{t\sim[d]}[\neg\text{useful}_Q(\boldsymbol{\mu}) \wedge \mu[i] \neq 0] \\
&\quad\leq \frac{\tau_\Delta}{b} + \mathsf{Pr}_{\boldsymbol{\mu}}[\neg\text{useful}_Q(\boldsymbol{\mu}) \wedge \mu[i] \neq 0] \\
&\quad\leq 2\frac{\tau_\Delta}{b},
\end{aligned} \tag{18}$$

where the first inequality is by our assumption on the predicate $\neg\text{useful}_Q(\boldsymbol{\mu}) \wedge \mu[i] \neq 0$ and the second inequality follows from $Q \in \mathcal{Q}(L)$ (by Definition A.3). Using logic, for $t \in [d]$:

$$(h(\boldsymbol{\mu}_t) \wedge \text{useful}_Q(\boldsymbol{\mu}_t)) \implies h(\boldsymbol{\mu}_t)$$

$$h(\boldsymbol{\mu}_t) \implies (h(\boldsymbol{\mu}_t) \wedge \text{useful}_Q(\boldsymbol{\mu}_t)) \vee (\neg\text{useful}_Q(\boldsymbol{\mu_t}) \wedge \mu[i] \neq 0)$$

Therefore using (18) we get

$$\mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)] - \frac{2}{b}\tau_\Delta \leq \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) \wedge \text{useful}_Q(\boldsymbol{\mu}_t)] \leq \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)] \tag{19}$$

We now use a proof by induction over queries issued to TM to establish that the output for each query vector $\boldsymbol{v}$ and key i is correct.

Suppose the output was correct until the current query. Then using Remark A.2, the set of current active buckets is a superset of the set of useful buckets for Q and a subset of all buckets. Therefore,

$$\mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) \wedge \text{useful}_Q(\boldsymbol{\mu}_t)] \leq \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) \wedge t \text{ is active}] \leq \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)] \tag{20}$$

Combining (19) and (20) we obtain

$$\mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)] - \frac{2}{b}\tau_\Delta \leq \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) \wedge t \text{ is active}] \leq \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)]$$

$$\implies \left|\mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)] - \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) \wedge t \text{ is active}]\right| \leq \frac{2}{b}\tau_\Delta \tag{21}$$

From our assumption on the predicate h we have

$$\left|\mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) - \mathsf{Pr}_{\boldsymbol{\mu}\sim\mathcal{B}}[h(\boldsymbol{\mu})]\right| \leq \frac{1}{b}\tau_\Delta \tag{22}$$

Therefore using (17), (21), and (22):

$$\begin{aligned}
&\left|\frac{1}{d}\hat{F} - \mathsf{Pr}_{\boldsymbol{\mu}\sim\mathcal{B}}[h(\boldsymbol{\mu})]\right| \\
&\leq \left|\frac{1}{d}\hat{F} - \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) \wedge t \text{ is active}]\right| \\
&+ \left|\mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) \wedge t \text{ is active}] - \mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t)]\right| \\
&+ \left|\mathsf{Pr}_{t\sim[d]}[h(\boldsymbol{\mu}_t) - \mathsf{Pr}_{\boldsymbol{\mu}\sim\mathcal{B}}[h(\boldsymbol{\mu})]\right| \\
&\leq \frac{4}{b}\tau_\Delta
\end{aligned}$$

Therefore the conditions of Lemma 3.4 hold and the reporting is correct: Only keys in $\mathrm{suspect}$ can be reported and all keys in heavy are reported. This finishes the induction step proof. □

In the oblivious setting, when the buckets are drawn i.i.d. and do not depend on Q, we can guarantee that the assumption holds with probability $1-\beta$ by choosing $d = O(\log(nm/\beta))$ and applying Chernoff and union bounds over predicates. In the following we analyze the adaptive setting.

A.5. Removing the assumption

We now establish that the assumption holds with probability at least $1-\beta/2$ using the generalization property of DP. We show that with an appropriate choice of constants and $Q \in \mathcal{Q}(L)$, all predicates in H are estimated well from the sample:

Lemma A.7. *Consider an execution of Algorithm 2 with adaptive inputs Q and i.i.d $(\boldsymbol{\mu}_t)_{t\in[d]}$. We show that with appropriate setting of constants and setting the dimension as in* (13)*, then as long as the adaptive query sequence satisfies $Q \in \mathcal{Q}(L)$ and $|Q| \le m$ then with probability $1-\beta/2$, for all the $n(4m+1)$ predicates $h \in H$,* (16) *holds.*

Proof. We apply Theorem A.2. We choose the parameter in the statement of the theorem to be $T \ge 2n(4m+1)/\beta$. With this setting and a union bound we get the bound with probability $1-\beta/2$ for all our $n(4m+1)$ predicates.

We treat Algorithm 2 and the "adversary" issuing the adaptive queries based on prior outputs as a single algorithm that operates on the measurement vectors $(\boldsymbol{\mu}_t)_{t\in[d]}$ and outputs queries $Q = (\boldsymbol{v}_q)$. The predicates in H are all computed from Q.

The algorithm provides (ξ,η)-differential privacy. From post processing, all the predicates H also preserve the privacy of $(\boldsymbol{\mu}_t)_{t\in[d]}$. Using Theorem A.4, we have

$$\xi = O(\sqrt{L}\varepsilon + L\varepsilon^2)$$
$$\eta = O(L\delta).$$

Recall that $\varepsilon = C_1/\sqrt{L}$. We choose the constant C_1 to be at most its value from Remark A.5 and as needed to set $\xi \le \tau_\Delta/6$.

Recall that $\delta = C_2/(m\cdot n\cdot b\cdot L)$. We choose C_2 similarly to provide $\eta \le \tau_\Delta/(6\cdot T\cdot b)$.

To summarize we have

$$T = \frac{2n(4m+1)}{\beta}$$
$$\eta \le \frac{\tau_\Delta}{6\cdot T\cdot b}$$
$$\xi \le \frac{\tau_\Delta}{6}$$
$$d \ge \frac{12b\log(T+1)}{\xi\tau_\Delta}$$

We now express the additive error on the estimate provided in Theorem A.2. The expected value of all our predicates is at most $1/b$. The additive contribution due to the multiplicative error term is therefore at most $(1-e^{-2\xi})/b \approx 2\xi/b$. The additional additive terms are $\frac{4}{\xi d}\log(T+1)$ and $2\cdot T\cdot\eta$. We can substitute our choices of T, d, ξ, η and obtain that each of these three terms is at most $\frac{1}{3b}\tau_\Delta$. Therefore the total additive error is $\frac{1}{b}\tau_\Delta$. □

B. Robust Weight Estimates

Our robust estimator (Algorithm 2) outputs a set $\mathtt{K}$ of $O(k)$ keys that includes all heavy hitters. In the following, we show that it is possible to also provide estimates for the weight of each key in $\mathtt{K}$ within the same asymptotic robustness guarantees.

Here we make a mild assumption for the input query vector $\boldsymbol{v} \in \mathbb{R}^n$. That is, we assume each entry of $\boldsymbol{v}$ is an integer of magnitude at most polynomial in n (say, n^c for a constant c).

One approach to provide such weight estimates is to use a private median algorithm (based on the median/quantile estimator reviewed in the preliminaries Section 2.3). There are suitable off-the-shelf private approximate median algorithms

(10; 14; 13; 33). However, this approach constitutes a different component that does not fit seamlessly in the framework of Algorithm 2. Instead, we design `WeightEstimator` (Algorithm 3) that implements a private median computation by accessing the buckets only through calls to the same threshold monitor TM that is initialized in Algorithm 2.

```
Algorithm 3: WeightEstimator
Input: A query vector $\boldsymbol{v} \in \mathbb{R}^n$, an index $i \in [n]$.
$(\boldsymbol{\mu}_t)_{t\in[d]} \leftarrow$ initialized BCountSketch with parameters $(n, d, b)$
TM $\leftarrow$ a ThresholdMonitor based on $(\boldsymbol{\mu}_t)_{t\in[d]}$.
for $w = -n^c, \ldots, 0, 1, \ldots, n^c$ do
    Define $f_{\leq w} : [d] \to \{0,1\}$ where $f_{\leq w}(t) = \mathbb{1}\{\mu_t[i] \neq 0 \wedge c_t \cdot \mu_t[i] \leq w\}$
    if TM.$query(f_{\leq w}, +1, \frac{d}{b}\tau_{\text{tr}}) = \top$ then
        return $\tilde{v}_i = w$
return $\tilde{v}_i = n^c$
```

Algorithm 3 uses the following parameters: $\tau_{\text{down}} = \frac{1}{10}$, $\tau_{\text{up}} = \frac{9}{10}$, and $\tau_{\text{tr}} = \frac{\tau_{\text{down}}+\tau_{\text{up}}}{2}$. Note that TM reports at most one "$\top$" for each key in K. Therefore, we can still get the same DP guarantee (by incurring a constant multiplicative factor, because we may need to have the "access limit" threshold L in the threshold monitor doubled[5]).

Efficiency. Implementing Algorithm 3 as described would be highly inefficient: it takes $\Theta(n^c)$ time to estimate a single key v_i. In the following, we describe an $\widetilde{O}(d)$ time implementation which has identical output distribution as Algorithm 3.

Our efficient implementation is based on a simple observation: let $w, w+1 \in [-n^c, n^c]$ be two consecutive integers. If for every $\boldsymbol{\mu}_t$ it holds that $\mu_t[i] \neq w$, then we can conclude that $f_{\leq w}$ and $f_{\leq w+1}$ are the same predicate. Therefore, when querying the threshold monitor with either $f_{\leq w}$ or $f_{\leq w+1}$, we have the same probability of getting $\top$. The same can be said for a range of potential weights: let $w_l < w_r$ be two integers. If for every $\boldsymbol{\mu}_i$ it holds that $\boldsymbol{\mu}_i \notin [w_l, w_r - 1]$, then $(f_{\leq w})$ for every $w \in [w_l, w_r - 1]$ refers to the same predicate, which means we have the same probability of getting $\perp$ when querying with $f_{\leq w}$ for every $w \in [w_l, w_r - 1]$.

We are ready to describe the implementation.

1. By the observation above, we know that the buckets $(\boldsymbol{\mu}_t)_{t\in[d]}$ partition the answer space $[-n^c, n^c]$ into $d+1$ intervals ($d/b + 1$ in expectation), where the same query predicate is used inside each interval. We scan these intervals from left to right.

2. For an interval $[L, R]$, we count the number of buckets such that $\mu_i[t] \leq L$ and calculate the probability that the threshold monitor reports $\top$ on any query in this range. Let p denote the probability. Conditioning on Algorithm 3 not returning before $w = L$, the probability that Algorithm 3 does not return an estimation before moving w to $R+1$ is $(1-p)^{R-L}$. With probability $(1-p)^{R-L}$, we proceed to the next interval.

3. Otherwise, we simulate the case that the algorithm returns an integer in the range $[L, R]$. We first calculate a normalizer $P = \sum_{w=1}^{R-L+1}(1-p)^{i-1} \cdot p$. Conditioning on Algorithm 3 returns an integer in $[L, R]$, it returns $w \in [L, R]$ with probability exactly $\frac{(1-p)^{w-L}p}{P}$. Therefore, we sample a real r uniformly in $[0, P]$, and find the smallest $w \in [L, R]$ such that $r < \sum_{j=L}^{w}(1-p)^{j-L} \cdot (p)$. We output w.

It is straightforward to verify that this implementation has the same output distribution as Algorithm 3.

Correctness. We analyze the correctness and privacy guarantee of Algorithm 3 now. We observe that `WeightEstimator` (Algorithm 3) fits well in the framework of Algorithm 2. That is, the weight estimator only accesses the buckets through TM (the threshold monitor). And TM reports at most one "$\top$" for each key in K. Therefore, we can still get the same DP guarantee (by incurring a constant multiplicative factor, because we may need to have the "access limit" threshold L in the threshold monitor doubled).

[5]Check Algorithm 1 for the definition of L.

Next, we will show that Algorithm 3 returns an estimate of $v[i]$ within additive error $\pm\frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2$ with high probability.

Let $\Gamma = O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\log\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)\log\frac{r}{\beta}\right)$ where the constant in the big-Oh is the constant in Theorem A.3. Then, by Theorem A.3, with probability at least $1-\beta$, Algorithm 3 returns an estimation $\tilde{v}_i$ such that:

- There are at least $\frac{d}{b}\tau_{\mathrm{tr}} - \Gamma$ active buckets μ_t where $\mu_t[i] \neq 0, \mu_t[i]\cdot c_t \leq \tilde{v}_t$.
- There are at most $\frac{d}{b}\tau_{\mathrm{tr}} + \Gamma$ active buckets μ_t where $\mu_t[i] \neq 0, \mu_t[i]\cdot c_t \leq \tilde{v}_t - 1$.

In the following, we use $\mathcal{E}_1$ to denote the event that both of the two conditions above hold.

Call a bucket $\boldsymbol{\mu}_t$ *good* for the estimation of $v[i]$, if both of the following hold:

1. $\boldsymbol{\mu}_t$ is active by the time we perform the estimation;
2. $\mu_t[i] \neq 0$ and $\mu_t[i]\cdot c_t \in v[i] \pm \frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2$.

Also, we call a bucket $\boldsymbol{\mu}_t$ *bad* if $\boldsymbol{\mu}_t$ is active, $\mu_t[i] \neq 0$ and $\mu_t[i]\cdot c_t \notin v[i] \pm \frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2$. We prove the following lemma.

Lemma B.1. *Consider the execution of Algorithm 3 for the vector* $\mathbf{v}$ *and index* i. *Suppose all of the following hold:*

- *The event* $\mathcal{E}_1$.
- *The number of good buckets is more than* $\frac{d}{b}\tau_{\mathrm{tr}} + \Gamma$.
- *The number of bad buckets is less than* $\frac{d}{b}\tau_{\mathrm{tr}} - \Gamma$.

Then, Algorithm 3 returns an estimation of $v[i]$ *within additive error* $\frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2$.

Proof. Define the multi-set $S_i = \{\boldsymbol{\mu}_t[i]\cdot c_t : \mu_t \text{ is active} \wedge \mu_t[i] \neq 0\}$. Call an element $x \in S_i$ good if $x \in v[i] \pm \frac{1}{\sqrt{k}}\|v_{\mathsf{tail}[k]}\|_2$. Let n_{good} denote the number of good elements in S_i. We call all other elements bad and let n_{bad} denote the number of such elements. We observe that n_{good} is equal to the number of good buckets and n_{bad} is equal to the number of bad buckets.

Let $\tilde{v}_i$ denote the output of the private median algorithm. Suppose $\tilde{v}_i$ is larger than or equal to n_1 elements in S_i. Since $\mathcal{E}_1$ holds, we have $n_1 \geq \frac{d}{b}\tau_{\mathrm{tr}} - \Gamma$. Because $n_{\mathrm{bad}} < \frac{d}{b}\tau_{\mathrm{tr}} - \Gamma$, we know those n_1 elements cannot be all bad. Namely, $\tilde{v}_i$ is no less than at least one good element.

Now, suppose $\tilde{v}_i - 1$ is larger than or equal to n_2 elements in S_i. Since $\mathcal{E}_1$ holds, we have $n_2 \leq \frac{d}{b}\tau_{\mathrm{tr}} + \Gamma$. Because $n_{\mathrm{good}} > \frac{d}{b}\tau_{\mathrm{tr}} + \Gamma$, we know those n_2 elements do not include all good elements. Namely, $\tilde{v}_i - 1$ is less than at least one good element.

Combining two observations, we conclude that $\tilde{v}_i$ is sandwiched between two good elements, which means $\tilde{v}_i \in v[i] \pm \frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|$ as desired. □

Now let us consider the oblivious setting. In this setting, we can fix an input sequence Q beforehand and analyze the behavior of our algorithm on it. In the next lemma, we will show that Algorithm 3 is accurate, so long as the number of good/bad buckets is close to its expectation (assuming the buckets are independent to the input Q).

Lemma B.2. *Consider an execution Algorithm 2 where it also runs Algorithm 3 to estimate the weight of* $v[i]$ *whenever it includes a key* i *in* K. *Fix the input sequence* $Q = (\boldsymbol{v}_q)_{q\in[m]} \in \mathcal{Q}(L)$ *and consider one query vector* $\boldsymbol{v} \in \mathcal{Q}(L)$. *Consider the following set of* $2n$ *predicates* $h : \mathrm{Supp}(\mathcal{B}) \to \{0,1\}$*:*

$$\begin{aligned} H = &\{\mathbb{1}_{\mu[i]\neq 0 \wedge \mu[i]\cdot c \in v[i] \pm \frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2 \wedge \mathrm{useful}_Q}\}_{i\in[n]} \\ &\cup \{\mathbb{1}_{\mu[i]\neq 0 \wedge \mu[i]\cdot c \notin v[i] \pm \frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2}\}_{i\in[n]} \ . \end{aligned}$$

If for all $h \in H$, it holds that the average of h over the buckets $(\boldsymbol{\mu}_t)_{t\in[d]}$ approximate well its expectation over $\boldsymbol{\mu} \sim \mathcal{B}$:

$$\left|\mathit{Pr}_{t\sim[d]} h(\boldsymbol{\mu_t}) - \mathit{Pr}_{\boldsymbol{\mu}\sim\mathcal{B}} h(\boldsymbol{\mu})\right| \le \frac{1}{b}\tau_\Delta. \tag{23}$$

Then Algorithm 3 always returns estimation for $v[i]$ within error $\pm\frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2$.

Before showing the proof, we observe that one can combine Lemma B.2 with a Chernoff bound to show that our algorithm is accurate in the oblivious setting.

Proof. First, we consider predicates of the form $h_{i,\text{bad}} = \mathbb{1}_{\mu[i]\neq 0 \wedge \mu[i]\cdot c \notin v[i] \pm \frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2}$. Fix an index i and consider $h_{i,\text{bad}}$. By Chebyshev's inequality, we have that

$$\mathsf{Pr}_{\boldsymbol{\mu}\sim\mathcal{B}}\left[|\mu[i]\cdot c - v[i]| \in \frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\| \;\middle|\; \mu[i]\neq 0\right] \ge \tau_{\text{up}}$$

provided $b \ge C\cdot k$ for some constant C. (The proof is similar to that of Lemma G.3). Recall $\tau_{\text{down}} = \frac{1}{10}$ and $\tau_{\text{up}} = \frac{9}{10}$. We have

$$\mathsf{Pr}_{\boldsymbol{\mu}\sim\mathcal{B}}[h_{i,\text{bad}}(\boldsymbol{\mu})] \le \frac{\tau_{\text{down}}}{b}.$$

We also consider predicates of the form $h_{i,\text{good}} = \mathbb{1}_{\mu[i]\neq 0 \wedge \mu[i]\cdot c \in v[i] \pm \frac{1}{\sqrt{k}}\|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2 \wedge \boldsymbol{\mu}\text{ is useful}}$. by Definition A.3, we know that

$$\mathsf{Pr}_{\boldsymbol{\mu}\sim\mathcal{B}}[\boldsymbol{\mu}\text{ is useful} \mid \mu[i]\neq 0] \ge 1-\tau_\Delta,$$

By union bound, we know that

$$\mathsf{Pr}_{\boldsymbol{\mu}\sim\mathcal{B}}[h_{i,\text{good}}(\boldsymbol{\mu})] \ge (\tau_{\text{up}} - \tau_\Delta)\frac{1}{b}.$$

Now, if the condition in the lemma statement holds, for every $i\in[n]$ we have

$$\mathsf{Pr}_{t\sim[d]}[h_{i,\text{good}}(\boldsymbol{\mu}_t)] \ge \frac{\tau_{\text{up}} - 2\tau_\Delta}{b}$$

and

$$\mathsf{Pr}_{t\sim[d]}[h_{i,\text{bad}}(\boldsymbol{\mu}_t)] \le \frac{\tau_{\text{down}} + \tau_\Delta}{b}.$$

We choose τ_Δ to be less than $\frac{1}{10}$. By doing so, the condition of Lemma B.1 is met, and Algorithm 3 can report estimations within the desired accuracy. This completes the proof. ☐

Next, we consider the real execution of Algorithm 3 in the adaptive setting. Knowing that $Q, \boldsymbol{v}$ is (ξ,η)-DP with respect to the buckets $(\boldsymbol{\mu}_t)_{t\in[d]}$, we might use the generalization property of DP (Theorem A.2) to argue that the number of good buckets is still close to its expectation in the independent case. So the condition of Lemma B.2 is still met with high probability. This step is analogous to the argument in Section A.5 and we do not repeat it here.

C. Query Efficiency

Our robust estimator as described in Algorithm 2 tests all n keys using the `ThresholdMonitor` in order to determine the returned set `K` of candidate heavy hitters. This is computationally inefficient. In the following, we show how we can combine our algorithm with an efficient non-robust heavy-hitters sketch to support heavy-hitter queries that are both efficient and robust.

First, let us recall the following fast oblivious heavy-hitter algorithm from (36):

Theorem C.1 ((36))**.** *There is an oblivious streaming algorithm* `FastQueryHH` *satisfying the following. For every $n, k \ge 1$ and $\beta \in (0,1)$,* `FastQueryHH` *maintains a vector $\boldsymbol{v}\in\mathbb{R}^n$ using $O(k\log n/\beta)$ space, supports entry updates in $O(\log n/\beta)$ time. On a query, it produces in $O(k\text{polylog}(n))$ time a list $L \subset [n]$ of indices such that:*

- *$|L| \le O(k)$.*

- *For every $i \in [n]$ such that $v[i] \geq \frac{1}{\sqrt{k}} \|\boldsymbol{v}_{\mathsf{tail}[k]}\|_2$, it holds that $i \in K$.*

With `FastQueryHH` in hand, we can slightly modify Algorithm 2 as follows: Concurrently to an execusion of Algorithm 2, we run a copy of `FastQueryHH` with the parameter "k" in Theorem C.1 set to $(C_a + 1) \cdot b$. Then on each heavy-hitter query, we first invoke `FastQueryHH` to produce a list L of candidate sets. Then we test each key $i \in L$ by querying the TM (just like what we have done in Algorithm 2). We call the new algorithm `FastQueryVariant`.

We analyze the query time and correctness of `FastQueryVariant`. On a heavy-hitter query, `FastQueryVariant` first takes $O(b \log n/\beta)$ time to run `FastQueryHH` and produces a list L. Then it tests each key from L by querying the `ThresholdMonitor`, which requires $O(b \cdot \ell)$ time for each key. Recall that $b = O(k)$. Therefore, the query time is bounded by $O(\mathrm{poly}(k, \ell, \log n/\beta))$. `FastQueryVariant` is also correct with high probability: the final list has size bounded by $O(k)$ and every heavy hitter will appear in the list L and then get selected by the `ThresholdMonitor` test with high probability.

Finally, we argue that `FastQueryVariant` can robustly answer at least $\widetilde{\Omega}(\ell^2)$ adaptively-chosen queries. This is because for every $i \in [n]$ with $|v_i| \leq \frac{1}{\sqrt{k}} \|\boldsymbol{v}_{\mathsf{tail}[C_a \cdot b]}\|$, it does not really matter if i is included in the list L or not, for such i will always be rejected by the `ThresholdMonitor` test with high probability. Therefore, the *statistical distance* between the output distribution of `FastQueryVariant` and that of Algorithm 2 is negligible. Hence, `FastQueryVariant` basically enjoys the same robustness guarantee as Algorithm 2, and the analysis of Algorithm 2 implies `FastQueryVariant` is also robust.

D. Robust Stable Estimator for Continuous Reporting

In this section we describe Algorithm 4, which is a robust version of the stable sign-alignment estimator presented in Section 3.2.

One important class of applications of `CountSketch` is on distributed or streaming data, which has the form of updates to weights of keys. In a streaming model the input is presented as a sequence of updates $(\boldsymbol{u}_j)_{j=1}^m$, where each update has the form $\boldsymbol{u} = (\boldsymbol{u}.\mathrm{key}, \boldsymbol{u}.\mathrm{value})$ such that $\boldsymbol{u}.\mathrm{key} \in [n]$ and $\boldsymbol{u}.\mathrm{value} \in \mathbb{R}$. The stream corresponds to an input vector $\boldsymbol{v}$ that is updated in each step. We can denote that as a sequence $(\boldsymbol{v}_q)$ of input vectors where $\boldsymbol{v}_q$ is the vector at step q. Initially, $\boldsymbol{v}_0 = 0^n$ and for step $q \geq 1$

$$\boldsymbol{v}_q := \boldsymbol{v}_{q-1} + u_q.\mathrm{value} \cdot \boldsymbol{e}_{u_q.\mathrm{key}} \ .$$

The sketch of streaming data can be efficiently maintained: Consider an initialized `BCountSketch` with measurement vectors $(\boldsymbol{\mu}_t)$. The sketch is initialized by setting $(c_t \leftarrow 0)_{t \in [d]}$ and updating $(c_t \leftarrow c_t + \mu_t[u.\mathrm{key}] \cdot u.\mathrm{value})_{t \in [d]}$.

Algorithm 4 is similar to the robust threshold estimator, but it only accounts for *changes* in the reporting (in terms of when access counts on the TM are increased). A change occurs when a key that is not currently reported enters the reported set or when a key that is currently reported exits the reported set.

Analysis: The analysis mimics that of Algorithm 2. The only differences is that we use $\tau_\Delta := (\tau_b - \tau_a)/25$, apply Lemma 3.5 for the quality of approximations needed, and define $\lambda_{Q,i}$ as the flip number (Definition 4.2) and use Remark 4.3 to establish that it bounds the number of TM queries returning $\top$ for the respective key. The definition of useful buckets, sequences $\mathcal{Q}(L)$, Lemma A.4, and Theorem A.1 all carry over.

Reporting estimated weights: The robust estimator in Algorithm 4 can optionally report estimated weights. When a key i enters the reported set, we obtain a private estimate $\hat{v}[i]$ of its current value, as described in Section B. While keys are in the reported set, we apply exact updates to their estimated values. When a key exits the reported set we delete its estimated value. As for the analysis, one TM query returning $\top$ is used upon an entry of a key to the reported set. Following that we perform exact updates, which do not affect the privacy and robustness analysis. The estimates we maintain however need to be validated as we seek accuracy with respect to the tail of the *current* $\boldsymbol{v}_q$ and the permitted "tail error" might decrease. This can make the estimate obtained earlier when key i entered the reported set not sufficiently accurate. Therefore, we validate our estimates before reporting them and perform an update (a fresh weight estimates) if the validation fails. The validation tests (using predicate queries to TM) if our maintained estimate is still within appropriate noisy quantiles of the bucket estimates. Each validation failure that triggers an update corresponds to a significant decrease of the tail error. This number contributes to the robustness budget when weights are reported.

Algorithm 4: Robust BCountSketch: Streaming

```
Input: Sketch parameters $(n, d, b)$, Access limit $L$, upper bound $m \geq L$ on the number of updates.
$(\boldsymbol{\mu}_t)_{t\in[d]} \leftarrow$ initialized BCountSketch with parameters $(n, d, b)$
$\tau_{m_1} \leftarrow \tau_a + \frac{1}{5}(\tau_b - \tau_a)$// lower threshold value for TM
$\tau_{m_2} \leftarrow \tau_b - \frac{1}{5}(\tau_b - \tau_a)$// higher threshold value for TM
$\mathtt{K}^+, \mathtt{K}^- \leftarrow \emptyset$// Initialize continuously reported set $\mathtt{K}^+ \cup \mathtt{K}^-$ of candidate heavy hitter keys
initialize ThresholdMonitor TM:
    TM.$S \leftarrow (\boldsymbol{\mu}_t)_{t\in[d]}$ // Measurement vectors
    TM.$(\varepsilon, \delta) \leftarrow \left(\frac{C_1}{\sqrt{L}}, \frac{C_2}{n\cdot m\cdot b\cdot L}\right)$// privacy parameters
    TM.$L \leftarrow L$ // access limit
foreach update $\boldsymbol{u} = (u.\text{key}, u.\text{value})$ do // process update
    // optional: If $u.\text{key} \in \mathtt{K}^+ \cup \mathtt{K}^-$ then $\hat{v}[u.\text{key}] += u.\text{value}$
    foreach $t \in [d]$ do // update sketch
        $c_t \leftarrow c_t + \mu_t[u.\text{key}] \cdot u.\text{value}$
    foreach $i \in [n] \setminus (\mathtt{K}^+ \cup \mathtt{K}^-)$ do // loop over keys not currently reported
        Define $f^+ : [d] \to \{0, 1\}$, where $f^+(t) = \mathbb{1}_{\mu_t[i]\cdot c_t > 0}$
        if TM.$query(f^+, \geq \frac{d}{b}\tau_{m2}) = \top$ then
            $\mathtt{K}^+ \leftarrow \mathtt{K}^+ \cup \{i\}$// Optional: $\hat{v}[i] \leftarrow$ estimate using Algorithm 3
        else
            Define $f^- : [d] \to \{0, 1\}$, where $f^-(t) = \mathbb{1}_{\mu_t[i]\cdot c_t < 0}$
            if TM.$query(f^-, \geq \frac{d}{b}\tau_{m2}) = \top$ then
                $\mathtt{K}^- \leftarrow \mathtt{K}^- \cup \{i\}$ // Optional: $\hat{v}[i] \leftarrow$ estimate using Algorithm 3
    foreach $\sigma \in \{+, -\}$ do // loop over keys currently reported
        foreach $i \in \mathtt{K}^\sigma$ do
            Define $f^\sigma : [d] \to \{0, 1\}$, where $f^\sigma(t) = \mathbb{1}_{\mu_t[i]\cdot c_t\cdot(\sigma 1) > 0}$
            if TM.$query(f^\sigma, \leq \frac{d}{b}\tau_{m1}) = \top$ then
                $\mathtt{K}^\sigma \leftarrow \mathtt{K}^\sigma \setminus \{i\}$ // Optional: delete $\hat{v}[i]$
    return $\{(i, \hat{v}[i]) \mid i \in \mathtt{K}^+ \cup \mathtt{K}^-\}$// Optional: test whether $\hat{v}[i]$ is still valid. If not replace $\hat{v}[i]$
        using Algorithm 3
```

E. Attack Strategy Overview

We describe our attack strategy which applies with both CountSketch and BCountSketch. The attacker knows the parameters of the sketch (n, d, b) but does not know the internal randomness $\boldsymbol{\rho}$ and thus the measurement vectors $(\boldsymbol{\mu}_t)_{t\in[d]}$. The attacker has an oracle access to the sketch realization by adaptively producing query vectors $(\boldsymbol{v}_q)$ and obtaining a set of keys:

$$\mathtt{K}(\boldsymbol{v}_q) := M(\mathtt{Sketch}_\rho(\boldsymbol{v}_q)).$$

We tailor our attack strategy to the following estimators M:

The median estimator: Compute for each key i the estimate $\hat{v}[i] \leftarrow \text{median}\{\mu_t[i]\cdot c_t \mid t \in T_i\}$ and return the k' keys with largest estimated magnitudes.

Basic threshold sign-alignment estimator: The estimator in Equation (5), with arbitrary τ, and with the basic estimate as in Equation (10).

Robust threshold sign-alignment estimator: As in Algorithm 2.

We establish lower bounds on robustness that nearly match the respective upper bounds:

Theorem E.1. *The median estimator and basic threshold sign-alignment estimator admit an attack of size $O(\ell)$. The robust estimator admits an attack of size $O(\ell^2)$, where $\ell = d/b$.*

For $i \in [n]$, we denote by $\boldsymbol{e}_i$ the unit vector of the i'th entry (that is, $e_i[j] = \mathbb{1}_{i=j}$). For a vector $\boldsymbol{z} \in \mathbb{R}^n$, the support of $\boldsymbol{z}$, which we denote by $\text{Supp}(\boldsymbol{z}) \subset [n]$, is the set of its nonzero entries. For a key $i \in [n]$, we denote its buckets by $T_i = \{t \in [d] \mid \mu_t[i] \neq 0\}$.

Attack Description Our attacks will be with respect to one designated key $h \in [n]$ and a specified bias-to-noise ratio parameter $\mathtt{BNR} > 0$. We construct an attack tail $\boldsymbol{a}$, which is a vector with entries in $\{-1, 0, 1\}$ so that $h \notin \text{Supp}(\boldsymbol{a})$ and support size $|\text{Supp}(\boldsymbol{a})| = r \cdot m$ (where r, m are attack parameters) that behaves very differently on the buckets T_h than on random buckets $\boldsymbol{\mu} \sim \mathcal{B}$. On a random bucket we have that $\mathbb{E}_{\boldsymbol{\mu}\sim\mathcal{B}}\left[\langle\boldsymbol{\mu}, \boldsymbol{a}\rangle\right] = 0$ and $\text{Var}_{\boldsymbol{\mu}\sim\mathcal{B}}\left[\langle\boldsymbol{\mu}, \boldsymbol{a}\rangle\right] = \frac{r\cdot m}{b}$ and standard deviation $\sqrt{\frac{1}{b}\cdot r\cdot m} = \sqrt{\frac{1}{b}}\|\boldsymbol{a}\|_2$. On most of the sketch buckets $(\boldsymbol{\mu}_t)_{t\in T_h}$, however, we will have large bias with

$$\langle\boldsymbol{\mu}_t, \boldsymbol{a}\rangle \cdot \mu_t[h] \geq \mathtt{BNR}\cdot\frac{1}{\sqrt{b}}\sqrt{r\cdot m}\,. \tag{24}$$

We then use $\boldsymbol{a}$ to design a vector on which the sketch fails. The details depend on the estimator. When considering inputs of the form

$$w \cdot \mathbf{e}_h \pm \boldsymbol{a}\,,$$

we can expect a randomly initialized sketch to provide a good estimate of w when $w \gg \sqrt{\frac{1}{b}\cdot r\cdot m} = \frac{1}{\sqrt{b}}\|\boldsymbol{a}\|_2$ but our attacked sketch masks w and the estimator might fail to return h even when $w \gg \frac{1}{\sqrt{k}}\|\mathsf{tail}_k(\boldsymbol{a})\|_2 \approx \sqrt{\frac{r\cdot m}{k}}$ or can return it as heavy even when $w = 0$.

Construction of $\boldsymbol{a}$ We construct our attack tail $\boldsymbol{a}$ by identifying and summing up $r = O(\ell)$ randomly-generated tails that are lightly biased in the same direction. When we sum r lightly-biased tails, the bias grows linearly with r whereas the standard deviation grows proportionally to $\sqrt{r}$. Therefore the ratio of the bias to the standard deviation increases.

We identify these lightly biased tails using queries in which h is a borderline heavy hitter and selecting tails on which h is reported (or selecting negated tails where it is not reported). We designate a special set H of keys that includes h and use query vectors of the form

$$\sum_{i\in H} v[i]\cdot\mathbf{e}_i \pm \boldsymbol{z}_q\,. \tag{25}$$

The values $v[i] \in \mathbb{R}_{\geq 0}$ for $i \in H$ are fixed throughout the attack.

The vectors $\boldsymbol{z}_q \in \{-1, 0, 1\}^n$ are chosen so that they have disjoint supports $\text{Supp}(\boldsymbol{z}_q)$ of size $|\text{Supp}(\boldsymbol{z})| = m$ and so that these supports are disjoint from H. For each key $i \in \text{Supp}(\boldsymbol{z}_q)$, the value $z_q[i]$ is drawn independently and uniformly at random from $\{-1, 1\}$. Our attack will process the sequence $(\boldsymbol{z}_q)$ and for each, we will either *collect* $\boldsymbol{z}_q$ (that is, $\boldsymbol{a} \leftarrow \boldsymbol{a} + \boldsymbol{z}_q$) collect its negation (that is, $\boldsymbol{a} \leftarrow \boldsymbol{a} - \boldsymbol{z}_q$) or collect neither. We stop after we collect a specified number r of tails.

Our attack on the median estimator uses $|H| = k' + 1$ with $k' - 1$ very heavy keys and two equal-weight borderline keys, one of which is h. The estimator outputs all very heavy keys and only one of the two borderline keys. If h is reported we collect $\boldsymbol{z}$ and if it is not reported we collect $-\boldsymbol{z}$.[6] For the sign-alignment estimators we use $H = \{h\}$. We query with both $\boldsymbol{z}$ and $-\boldsymbol{z}$. If h is reported exactly once then either $\boldsymbol{z}$ or its negation are collected accordingly. The querying with the robust estimator, which is probabilistic, requires $O(\ell)$ queries and is discussed below.

Analysis The contribution of each $\boldsymbol{z}$ to a bucket in T_h is a symmetric random variable that is the sum of independent random variables that are 0 with probability $(1 - 1/b)$ and 1 or -1 each with probability $1/2b$. The distribution has expectation 0 and standard deviation $\sqrt{m/b}$. We now consider the signs of the tail contributions $\mu_t[h]\langle \boldsymbol{\mu}_t, \boldsymbol{z}\rangle$ on all ℓ buckets (we use m that is large enough so a contribution of 0 is unlikely). Then each bucket is positive or negative with probability 0.5. Using an anti-concentration bound, $\boldsymbol{z}$ with constant probability would have $\Omega(\sqrt{\ell})$ difference between the number of positive or negative contributions. Note that when we condition on the sign of $\mu_t[h]\langle \boldsymbol{\mu}_t, \boldsymbol{z}\rangle$, the distribution of magnitudes is the same as with an unconditioned bucket.

We now assume we can identify which one of $\pm\boldsymbol{z}$ has more positive contributions and collect it. Then with constant probability its contribution to a given bucket in T_h is positive with probability $1/2 + \Omega(1/\sqrt{\ell})$ and negative otherwise (the distribution of magnitudes conditioned on the sign is the same as without conditioning on the selection). The process is equivalent as drawing a positive value with probability $\Omega(1/\sqrt{\ell})$ and an unbiased value (distributed as with random $\boldsymbol{z}$) otherwise. So with constant probability the collected vector adds in expectation bias of $\sqrt{m/(b \cdot \ell)}$ for each bucket of T_h and variance that is roughly that of an unconditioned bucket $\frac{m}{b}$. The expected bias added to a bucket after r collection events is then $\Omega(r \cdot \sqrt{m/b}/\sqrt{\ell})$ (which is also well concentrated when $r \gg \sqrt{\ell}$, since in each round there is $\Omega(1/\sqrt{\ell})$ probability of bias that is $\Omega(\sqrt{m/b})$). The variance of the unbiased part is bounded from above by that of adding tails $\boldsymbol{z}$ unconditionally, and is $\leq r \cdot m/b$.

In terms of the "tail norm." Each collection event contributes m entries of magnitude 1 to $\boldsymbol{a}$. Hence, after r collection events we have $\|\boldsymbol{a}\|_2 = \sqrt{r \cdot m}$. Therefore, the standard deviation of the per-bucket noise is $\approx \sqrt{r \cdot m/b} = \frac{1}{\sqrt{b}}\|\boldsymbol{a}\|_2$).

The number r of collection events needed for the bias-to-noise ratio that exceed a specified $\mathtt{BNR}$ is (up to constants) the solution of

$$r \cdot \sqrt{\frac{m}{b \cdot \ell}} = \mathtt{BNR} \cdot \sqrt{\frac{r \cdot m}{b}} ,$$

which yields $r = c \cdot \mathtt{BNR}^2 \ell$ (for some constant c).

What remains is to show how we set up the query vectors with tail $\pm\boldsymbol{z}$ so that we can identify lightly-biased tail.

Choosing the weight of key h We set the value of $v[h]$ so that collection happens with constant probability for each $\boldsymbol{z}_q$ and that the collected tail is biased in the "right" direction. To do so, we set $v[h]$ so that key h is "borderline" reported. For the median and basic threshold estimators, we set it so that over the distribution of $\boldsymbol{z}$ there is constant probability to report h and constant probability not to report h.

The robust estimator adds random noise to the count and has a probabilistic output. There is also differences in the count due to buckets getting inactivated during the execution. So for each count (minus the contribution of inactive buckets) there is a reporting probability. Recall that we can expect a gap of $\Omega(\sqrt{\ell})$ between the counts of $\boldsymbol{z}$ and $-\boldsymbol{z}$. The reporting probability is 0 for low counts and 1 for high counts (close to ℓ). Therefore there is a value where the reporting probability increases by at least $\Theta(1/\sqrt{\ell})$ when the count increases by $\Theta(\sqrt{\ell})$. Note that both the number of inactivated buckets and the DP noise added by TM are of a lower order that the count gap ($\Theta(1/\ell)$ fraction versus $\Theta(1/\sqrt{\ell})$ gap). Therefore the probability gap holds even when accounting to the DP noise and inactive buckets.

We set $v[h]$ to be in this regime. We can then use $O(\ell)$ queries with each of $\boldsymbol{z}$ and $-\boldsymbol{z}$ to estimate the probabilities to the needed accuracy (additive error of $O(1/\sqrt{\ell})$) to identify the larger one.

The attacks on the classic and basic estimators use $O(1)$ queries for each collection event so overall the attack uses $O(\ell)$

[6]We assume here that it is unlikely that the median estimates of the two borderline keys are equal. This can be assured with $\sqrt{m/b} \gg d/b$ or by using Gaussian values instead of $-1, 1$.

queries. The attack on the robust estimator requires $O(\ell)$ queries for each collection event and therefore overall uses $O(\ell^2)$ queries.

Extensions The attacks can be implemented in a streaming model in which case the size of the attack is replaced by the flip number λ_Q. We emulate our attacks using updates, noting that $2m$ updates are needed to change between a tail and its negation or between different tails. This while maintaining stability of the reporting of keys in H (by increasing their values during the updates of the tail).

The attacks can be generalized to the case when the parameters of the estimators are not known to the attacker. In this case, we include a "search" for the parameters as part of the attack.

F. Empirical Evaluation of `CountSketch` Variants and Estimators

We introduced `BCountSketch` in order to facilitate the robustness analysis. We also introduced the novel sign-alignment estimators. The two sketch variants `CountSketch` and `BCountSketch` and the estimators median and sign-alignment have similar asymptotic performance. We report here results from an empirical evaluation. The evaluation used input vectors with one "heavy" entry and n i.i.d. entries $N(0,1)$. The goal was to identify uniquely the heavy key. To do this with the median estimator, we computed median estimates for all keys and returned the key with the largest estimate. To do this with sign-alignment, we returned the key with the largest alignment. Code for our experiments can be found at `https://github.com/google-research/google-research/tree/master/robust_count_sketch`.

The results, reported in Figure 2 (sweeping value of heavy key) and Figure 3 (sweeping size d/b), show that performance of the different sketch-estimator pairs is similar: There is no considerable performance loss from `BCountSketch` over `CountSketch` (that arises because the number of estimates available for each entry is in expectation d/b with `BCountSketch` and exactly d/b with `CountSketch`). We also observed no considerable loss from using sign-alignment estimates (that offer different benefits, but here is evaluated on identifying the top value).

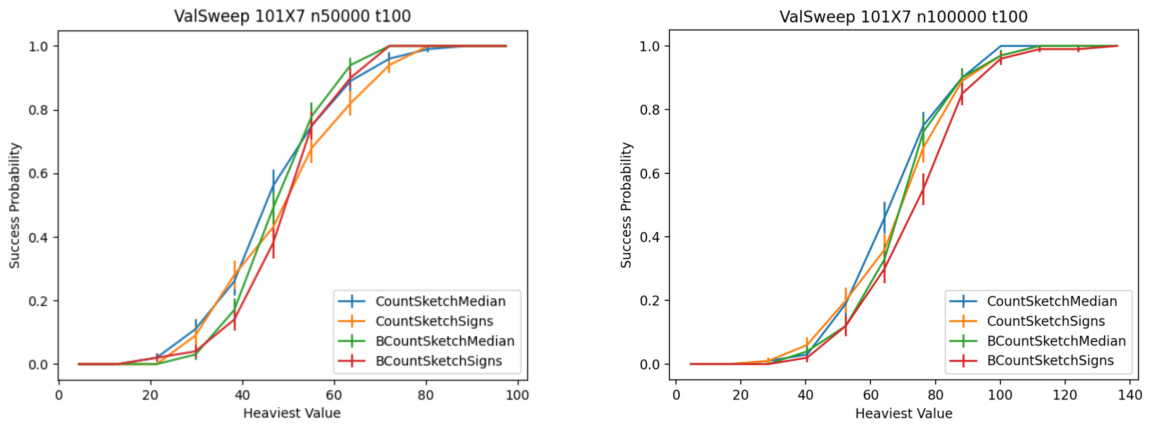

Figure 2. The y-axis shows the fraction of repetitions (out of 100) in which the heavy key was successfully returned, with error bars depicting one standard deviation. In both sub-figures the x-axis is the value v of the heavy entry. Sketch parameters $b = 7$, $d/b = 101$, left: $n = 5 \times 10^4$, right: $n = 1 \times 10^5$.

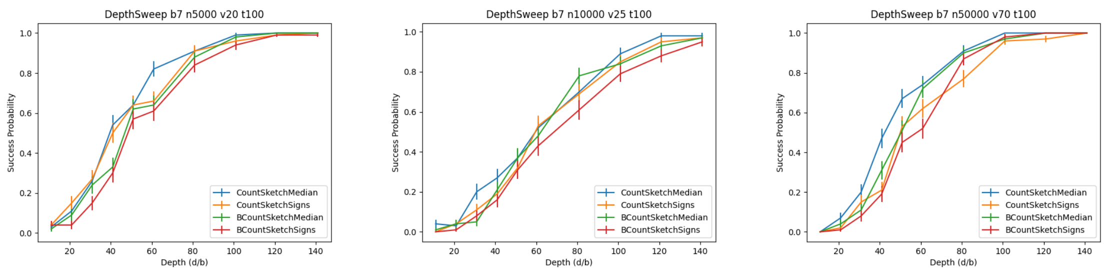

Figure 3. The y-axis shows the fraction of repetitions (out of 100) in which the heaviest key with value v was successfully returned, with error bars depicting one standard deviation. The x-axis is (d/b). We use $b = 7$. Left: $v = 20$, $n = 5 \times 10^3$ middle: $v = 25$, $n = 1 \times 10^4$ right: $v = 70$, $n = 5 \times 10^4$.

G. Proof of Lemma 3.1

In this section we provide the proof of Lemma 3.1. We start with some preliminaries.

G.1. Helpful Inequalities

Theorem G.1. *(Hölder's inequality) Let X, Y be two non-negative random variables. Also let $p, q \geq 1$ be two reals such that $\frac{1}{p} + \frac{1}{q} = 1$. Then it holds that $\mathbb{E}[X \cdot Y] \leq \|X\|_p \cdot \|Y\|_q := \mathbb{E}[X^p]^{1/p} \cdot \mathbb{E}[Y^q]^{1/q}$.*

Lemma G.1. *(technical lemma) Let $A, B \geq 0$ be two reals. Let X be a random variable such that:*

$$\mathbb{E}[X] = 0, \quad \mathbb{E}[X^2] \geq A,$$
$$\mathbb{E}[X^3] = 0, \quad \mathbb{E}[X^4] \leq B.$$

Then, for any $C \geq 0$ one has

$$Pr[X \leq -C] \geq \frac{1}{(16 \cdot (6AC^2 + B + C^4))^{1/3}} \left(\left(\frac{(A + C^2)^3}{6AC^2 + B + C^4} \right)^{1/2} - C \right)^{4/3}.$$

Proof. Define a new random variable $Y = X + C$. It suffices to bound $\Pr[Y \leq 0]$. First of all, one can verify the following useful moment bounds of Y.

$$\mathbb{E}[Y] = C,$$
$$\mathbb{E}[Y^2] \geq A + C^2,$$
$$\mathbb{E}[Y^4] \leq 6AC^2 + B + C^4.$$

Then, we have

$$\mathbb{E}[|Y| \cdot \mathbb{1}_{Y>0}] - \mathbb{E}[|Y| \cdot \mathbb{1}_{Y \leq 0}] = C. \tag{26}$$

We also have

$$\begin{aligned} \mathbb{E}[Y^2] &= \mathbb{E}[|Y|^{2/3} \cdot |Y|^{4/3}] \\ &\leq \left\| |Y|^{2/3} \right\|_{3/2} \cdot \left\| |Y|^{4/3} \right\|_3 \quad \text{(Theorem G.1)} \\ &= \mathbb{E}[Y]^{2/3} \cdot \mathbb{E}[Y^4]^{1/3}, \end{aligned}$$

which implies that

$$\mathbb{E}[|Y|] = \mathbb{E}[|Y| \cdot \mathbb{1}_{Y>0}] + \mathbb{E}[|Y| \cdot \mathbb{1}_{Y \leq 0}] \geq \left(\frac{\mathbb{E}[Y^2]^3}{\mathbb{E}[Y^4]} \right)^{1/2}. \tag{27}$$

Subtracting (26) from (27), one gets

$$\mathbb{E}[|Y| \cdot \mathbb{1}_{Y \leq 0}] \geq \frac{1}{2}\left(\left(\frac{\mathbb{E}[Y^2]^3}{\mathbb{E}[Y^4]}\right)^{1/2} - C\right). \tag{28}$$

On the other hand, we also have (by Theorem G.1)

$$\mathbb{E}[|Y| \cdot \mathbb{1}_{Y \geq 0}] \leq \|Y\|_4 \cdot \|\mathbb{1}_{Y \leq 0}\|_{4/3} \leq \mathbb{E}[Y^4]^{1/4} \cdot \mathsf{Pr}[Y \leq 0]^{3/4}. \tag{29}$$

Combining (28) and (29), one concludes

$$\mathsf{Pr}[Y \leq 0] \geq \frac{1}{2 \cdot (2 \cdot \mathbb{E}[Y^4])^{1/3}}\left(\left(\frac{\mathbb{E}[Y^2]^3}{\mathbb{E}[Y^4]}\right)^{1/2} - C\right)^{4/3}$$

as desired. □

In our analysis, we will instantiate Lemma G.1 with the following parameter setting.

Corollary G.2. *Let $A > 0$ be a real. Let X be a random variable such that:*

$$\mathbb{E}[X] = 0, \quad \mathbb{E}[X^2] = A,$$
$$\mathbb{E}[X^3] = 0, \quad \mathbb{E}[X^4] \leq 3A^2.$$

Then, it holds that $\mathsf{Pr}\left[X \leq -\frac{\sqrt{A}}{5}\right] \geq 0.02$.

Proof. By Lemma G.1 and direct calculation. □

G.2. Lemma proof details

We first recall how a linear measurement of a bucket is sampled. A bucket $\mu\colon [n] \to \{-1, 0, 1\}$ is constructed as follows: first, sample a 3-wise independent function $h\colon [n] \to \{0, 1\}$ such that for every $i \in [n]$, $h(i) = 1$ with probability $1/b$. Then, sample a 5-wise independent function $\sigma\colon [n] \to \{-1, 1\}$ such that $\sigma(i)$ equals 1 or -1 with equal probability. Finally define $\mu[i] = h(i) \cdot \sigma(i)$. Fix an input vector $\boldsymbol{v} \in \mathbb{R}^n$ and compute $c = \langle \boldsymbol{\mu}, \mathbf{v}\rangle$. The measurement yields the weak estimates $\hat{v}[i] := \mu[i]c$ of $v[i]$ for all $i \in [n]$ that participate in the bucket (that is, $\mu[i] \neq 0$). [7]

The first part is established by Lemma G.3 and the second part by Lemma G.4.

Lemma G.3. *If $|v[i]| \geq \frac{30}{\sqrt{b}}\|\boldsymbol{v}_{\mathsf{tail}[b/900]}\|_2$, then $\mathsf{Pr}[c \cdot \mu[i] \cdot \operatorname{sign}(v[i]) > 0 \mid \mu[i] \neq 0] \geq \frac{399}{400}$.*

Proof. We prove for the case that $\operatorname{sgn}(v[i]) = 1$. The case for $\operatorname{sgn}(v[i]) = -1$ can be handled similarly. Let

$$S = \{i \in [n] : |v[i]| \text{ is among the } \frac{b}{900} \text{ largest values in the multiset } \{|v[j]|\}_{j \in [n]}\}.$$

Conditioning on $\mu[i]$, we argue as follows. Let $c_{-i} = \sum_{i' \neq i, i' \notin S} \mu[i'] \cdot v[i']$. Then one has

$$\mathbb{E}_{\boldsymbol{\mu}}[c_{-i} \mid \mu[i] \neq 0] = 0,$$
$$\mathbb{E}_{\boldsymbol{\mu}}[c_{-i}^2 \mid \mu[i] \neq 0] \leq \frac{1}{b}\|\boldsymbol{v}_{\mathsf{tail}[b/900]}\|_2^2.$$

Let $\mathcal{E}_1$ denote the event that $\mu[i] \cdot c_{-i} < -\frac{30}{\sqrt{b}} \cdot \|\boldsymbol{v}_{\mathsf{tail}[b/900]}\|_2$. By Chebyshev's inequality one has

$$\mathsf{Pr}_{\boldsymbol{\mu}}[\neg\mathcal{E}_1 \mid \mu[i] \neq 0] = \mathsf{Pr}_{\boldsymbol{\mu}}\left[\mu[i] \cdot c_{-i} < -\frac{30}{\sqrt{b}} \cdot \|\boldsymbol{v}_{\mathsf{tail}[b/900]}\|_2 \;\middle|\; \mu[i] \neq 0\right] \leq \frac{1}{900}.$$

[7]The Lemma also holds for `CountSketch`. Note that while buckets of `CountSketch` can be dependent, the d/b buckets that a key i participates in are independent and follow the same distribution as `BCountSketch` buckets. The only difference is that we have exactly d/b buckets instead of in expectation.

Let $\mathcal{E}_2$ denote the event that $\mu[i'] = 0$ for all $i' \in S \setminus \{i\}$. We first observe that

$$\mathop{\mathbb{E}}_{\boldsymbol{\mu}}\left[\sum_{i' \in S\setminus\{i\}} \mathbb{1}_{\mu[i'] \neq 0} \;\middle|\; \mu[i] \neq 0\right] \le \frac{|S|}{b} \le \frac{1}{900}.$$

By Markov's inequality, we have

$$\mathsf{Pr}_{\boldsymbol{\mu}}\left[\neg\mathcal{E}_2 \mid \mu[i] \neq 0\right] = \mathsf{Pr}_{\boldsymbol{\mu}}\left[\sum_{i' \in S\setminus\{i\}} \mathbb{1}_{\mu[i'] \neq 0} \ge 1\right] \le \frac{1}{900}.$$

Note that when both $\mathcal{E}_1$ and $\mathcal{E}_2$ hold, we must have $c \cdot \mu[i] = \mu[i] \cdot (c_{-i} + v[i]) > 0$. Therefore, we conclude that

$$\mathsf{Pr}_{\boldsymbol{\mu}}\left[c \cdot \mu[i] > 0 \mid \mu[i] \neq 0\right] \ge 1 - \mathsf{Pr}_{\boldsymbol{\mu}}\left[\neg\mathcal{E}_1 \vee \neg\mathcal{E}_2 \mid \mu[i] \neq 0\right] \ge \frac{399}{400}.$$

□

Then, we show that if a key is far away from being heavy, then there is a decent chance that the sign of a key does not align with the sign of buckets it participates in. Let $i \in [n]$ be a key. Suppose there are at least $50b$ keys j such that $|v[j]| \ge |v[i]|$. Draw a random bucket $\boldsymbol{\mu}$ conditioning on $\mu[i] \neq 0$. For intuition, let us first consider the case that each coordinate of $\boldsymbol{\mu}$ is *independently* sampled. Then with probability 50% there exists $j \in [n]$ such that $|v[j]| \ge |v[i]|$ and $\mu(j) \neq 0$. Conditioning on this, with probability $\frac{1}{4}$ we have both $\operatorname{sign}(\mu(j) \cdot v[j]) \neq \operatorname{sign}(\mu(i) \cdot v[i])$ and $\operatorname{sign}\left(\sum_{j' \in [n]\setminus\{j,i\}} \mu(j') \cdot v[j']\right) \neq \operatorname{sign}(\mu(i) \cdot v[i])$. In this case, the sign of the bucket is "pushed away" from the direction of $v[i]$. Exploiting moment methods and Lemma G.1, we can show a similar conclusion even when $\boldsymbol{\mu}$ is sampled from a limited-independence source. See the lemma below.

Lemma G.4. *For every $i \in [n]$, if there are at least $50b$ keys j such that $|v[j]| \ge |v[i]|$, then it holds $\mathsf{Pr}_{\boldsymbol{\mu}}[c \cdot \mu[i] > 0 \mid \mu[i] \neq 0] \le 59/60$ and $\mathsf{Pr}_{\boldsymbol{\mu}}[c \cdot \mu[i] < 0 \mid \mu[i] \neq 0] \le 59/60$.*

Proof. We assume the vector $\boldsymbol{v}$ is sorted in decreasing order of absolute values. Namely, $|v[1]| \ge |v[2]| \ge \cdots \ge |v[n]|$, and we are proving for an item with index $i > 50k$. This assumption is only for the ease of presentation.

W.L.O.G. we condition on $\mu[i] = 1$. A similar argument gives the same bound for the case of $\mu[i] = -1$, and one can use the tower rule over two cases and conclude the proof. From now on, we will always condition on $\mu[i] = 1$.

Let $c_{-i} = \sum_{i' \neq i} \mu[i'] \cdot v[i]$. We want to show lower bounds for $\mathsf{Pr}[c_{-i} \le -|v[i]|]$ and $\mathsf{Pr}[c_{-i} \ge |v[i]|]$. In the following we only lower bound $\mathsf{Pr}[c_{-i} \le -|v[i]|]$. The other case is analogous.

Let us define a random variable $X = \sum_{j=1}^{50 \cdot b} \mathbb{1}_{\mu[j] \neq 0}$. That is, we consider the $50 \cdot k$ keys of largest magnitude and let X specify how many among them are hashed into the bucket. Then one has

$$\mathbb{E}[X] = 50, \quad \operatorname{Var}[X] \le 50.$$

Let $\mathcal{E}$ denote the event that $X \ge 25$. Applying a Chebyshev tells us that $\mathsf{Pr}[\mathcal{E}] \ge 1 - \frac{50}{2500} > \frac{9}{10}$.

Note that X and $\mathcal{E}$ depend only on h. Hence, conditioning on $\mathcal{E}$, h and $\mu[i] = 1$, the function σ is still 4-wise independent. Now let us consider c_{-i} (which is now a random variable depending on σ). In the following, we consider $\mathbb{E}_\sigma[c_{-i}^p \mid \mathcal{E}, h]$ for $p \in \{1, 2, 3, 4\}$. First, for $p = 1$ we have

$$\mathop{\mathbb{E}}_{\sigma}[c_{-i} \mid \mathcal{E}, h] = \sum_{i' \neq i} v[i] \cdot \mathop{\mathbb{E}}_{\sigma}[\mu[i']] = 0.$$

For $p = 2$ we have

$$\begin{aligned}\mathop{\mathbb{E}}_{\sigma}[c_{-i}^2 \mid \mathcal{E}, h] &= \sum_{i_1, i_2 \in [n]\setminus\{i\}} v[i_1] \cdot v[i_2] \cdot \mathop{\mathbb{E}}_{\sigma}[\mu[i_1]\mu[i_2]] \\ &= \sum_{i_1 \in [n]\setminus\{i\}} v[i_1]^2 \cdot \mathbb{1}_{h(i_1) \neq 0} \\ &\ge 25 \cdot v[i]^2.\end{aligned}$$

The second line is due to $\sigma(i_1)$ and $\sigma(i_2)$ are independent for $i_1 \neq i_2$. The last line is due to our assumption that $X \geq 25$. Then, for $p = 3$ we have

$$\mathop{\mathbb{E}}_{\sigma}[c_{-i}^3 \mid \mathcal{E}, h] = \sum_{i_1,i_2,i_3 \in [n]\setminus\{i\}} v[i_1] \cdot v[i_2] \cdot v[i_3] \cdot \mathop{\mathbb{E}}_{\sigma}[\mu[i_1]\mu[i_2]\mu[i_3]] = 0.$$

The second equality holds because for every tuple (i_1, i_2, i_3), there must be an index $i \in [n]$ that appears for *odd* times among i_1, i_2, i_3. Suppose i appears $t \in \{1, 3\}$ times. We observe that $\mathbb{E}_{\sigma}[\mu[i]^t] = 0$. Since σ is 3-wise independent, it follows that $\mathbb{E}_{\sigma}[\mu[i_1] \cdot \mu[i_2] \cdot \mu[i_3]] = \mathbb{E}[\mu[i]^t] = 0$. Finally, we consider $p = 4$, we have:

$$\mathop{\mathbb{E}}_{\sigma}[c_{-i}^4 \mid \mathcal{E}, h] = \sum_{i_1,i_2,i_3,i_4 \in [n]\setminus\{i\}} v[i_1] \cdot v[i_2] \cdot v[i_3] \cdot v[i_4] \cdot \mathop{\mathbb{E}}_{\sigma}[\mu[i_1]\mu[i_2]\mu[i_3]\mu[i_4]].$$

Following a similar reasoning as we have done above, we observe that a tuple (i_1, i_2, i_3, i_4) may have non-zero contribution, only if there are no indices that appear odd times among (i_1, i_2, i_3, i_4). Then, at least one of the following three conditions must hold: (1) $i_1 = i_2, i_3 = i_4$, (2) $i_1 = i_3, i_2 = i_4$ or (3) $i_1 = i_4, i_2 = i_3$. Therefore, we have:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma}[c_{-i}^4 \mid \mathcal{E}, h] &\leq 3 \cdot \sum_{i_1,i_2 \in [n]\setminus\{i\}} v[i_1]^2 \cdot v[i_2]^2 \cdot \mathop{\mathbb{E}}_{\sigma}[\mu[i_1]^2\mu[i_2]^2] \\
&\leq 3 \sum_{i_1,i_2 \in [n]\setminus\{i\}} v[i_1]^2 \cdot v[i_2]^2 \mathbb{1}_{h(i_1)\neq 0} \mathbb{1}_{h(i_2)\neq 0} \\
&\leq 3 \left(\sum_{i_1 \in [n]\setminus\{i\}} v[i_1]^2 \cdot \mathbb{1}_{h(i_1)\neq 0} \right)^2 \\
&\leq 3 \mathop{\mathbb{E}}_{\sigma}[c_{-i}^2 \mid \mathcal{E}, h]^2.
\end{aligned}$$

In summary, we have the following.

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\boldsymbol{\mu}}[c_{-i} \mid \mathcal{E}, h] = \mathop{\mathbb{E}}_{\boldsymbol{\mu}}[{c_{-i}}^3] = 0, \\
&\mathop{\mathbb{E}}_{\boldsymbol{\mu}}[{c_{-i}}^2 \mid \mathcal{E}, h] \geq 25 v[i]^2, \\
&\mathop{\mathbb{E}}_{\boldsymbol{\mu}}[{c_{-i}}^4 \mid \mathcal{E}, h] \leq 3 \cdot \mathbb{E}[{c_{-i}}^2 \mid \mathcal{E}, h]^2.
\end{aligned}$$

By Corollary G.2, it follows that

$$\mathsf{Pr}[c_{-i} \leq -v[i] \mid \mathcal{E}, h] \geq \frac{2}{100}.$$

Therefore, we have

$$\mathsf{Pr}_{h,\sigma}[c_{-i} \leq -v[i]] \geq \mathsf{Pr}_h[\mathcal{E}] \cdot \mathsf{Pr}[c_{-i} \leq -v[i] \mid \mathcal{E}, h] \geq \frac{2}{100} \cdot \frac{9}{10} \geq \frac{1}{60},$$

as desired. ☐

Corollary G.5. *Let $\boldsymbol{v} \in \mathbb{R}^n$ be a vector. For every $i \in [n]$, if $|v[i]| \leq \|\boldsymbol{v}_{\mathsf{tail}[50b]}\|$, then $\mathsf{Pr}_{\boldsymbol{\mu}}[c \cdot \mu[i] > 0 \mid \mu[i] \neq 0] \leq 59/60$ and $\mathsf{Pr}_{\boldsymbol{\mu}}[c \cdot \mu[i] < 0 \mid \mu[i] \neq 0] \leq 59/60$.*

Proof. $|v[i]| \leq \|\boldsymbol{v}_{\mathsf{tail}[50b]}\|$ implies $|v[i]|$ is less than or equal to at least $50b$ other $v[j]$'s. The conclusion follows from Lemma G.4 immediately. ☐

To wrap-up, we combine Lemma G.3 and Corollary G.5 to justify Lemma 3.1 with constants $C_a = 50, C_b = 30, \tau_a = \frac{59}{60}$ and $\tau_b = \frac{399}{400}$.