Space-optimal Heavy Hitters with Strong Error Bounds

Radu Berinde
MIT
texel@mit.edu

Graham Cormode
AT&T Labs–Research
graham@research.att.com

Piotr Indyk*
MIT
indyk@mit.edu

Martin J. Strauss†
University of Michigan
martinjs@umich.edu

ABSTRACT

The problem of finding heavy hitters and approximating the frequencies of items is at the heart of many problems in data stream analysis. It has been observed that several proposed solutions to this problem can outperform their worst-case guarantees on real data. This leads to the question of whether some stronger bounds can be guaranteed. We answer this in the positive by showing that a class of "counter-based algorithms" (including the popular and very space-efficient FREQUENT and SPACESAVING algorithms) provide much stronger approximation guarantees than previously known. Specifically, we show that errors in the approximation of individual elements do not depend on the frequencies of the most frequent elements, but only on the frequency of the remaining "tail." This shows that counter-based methods are the most space-efficient (in fact, space-optimal) algorithms having this strong error bound.

This tail guarantee allows these algorithms to solve the "sparse recovery" problem. Here, the goal is to recover a faithful representation of the vector of frequencies, f. We prove that using space $O(k)$, the algorithms construct an approximation f^* to the frequency vector f so that the L1 error $\|f - f^*\|_1$ is close to the best possible error $\min_{f'} \|f' - f\|_1$, where f' ranges over all vectors with at most k non-zero entries. This improves the previously best known space bound of about $O(k \log n)$ for streams without element deletions (where n is the size of the domain from which stream elements are drawn). Other consequences of the tail guarantees are results for skewed (Zipfian) data, and guarantees for accuracy of merging multiple summarized streams.

*Supported in part by David and Lucille Packard Fellowship and by MADALGO (Center for Massive Data Algorithmics, funded by the Danish National Research Association) and by NSF grant CCF-0728645.

†Supported by NSF CAREER award CCF 0743372 and DARPA/ONR N66001-08-1-2065

PODS'09, June 29–July 2, 2009, Providence, Rhode Island, USA.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

frequency estimation, heavy hitters, streaming algorithms

1. INTRODUCTION

Data stream algorithms have become an indispensable tool for analyzing massive data sets [21, 27]. Such algorithms aim to process huge streams of updates in a single pass and store a compact summary from which properties of the input can be discovered, with strong guarantees on the quality of the result. This approach has found many applications, in large scale data processing and data warehousing [19, 4, 16, 18], as well as in other areas, such as network measurements [1, 11, 13, 15], sensor networks [5, 29] and compressed sensing [17, 7].

Finding the "heavy hitters" is one of the quintessential problems in data stream algorithms. Given a stream of items (possibly with weights attached), find those items with the greatest total weight. This is an intuitive problem, that applies to many natural questions: given a stream of search engine queries, which are the most frequently occurring terms? Given a stream of supermarket transactions and prices, which items have the highest total dollar sales? Further, this simple question turns out to be a core subproblem of many more complex computations over data streams, such as estimating the entropy [8], and clustering geometric data [20]. Therefore, it is of high importance to design efficient algorithms for this problem, and understand the performance of existing ones.

The problem can be formalized into one of estimating item frequencies. In this problem we are given a stream of N elements from some universe; the goal is to compute, for each universe element i, an estimator $\hat{f}_i$ that approximates f_i, the number of times the element i occurs in the data stream (or the sum of associated weights in a weighted version). Such estimators provide a succinct representation of the data stream, with a controllable trade-off between description size and approximation error.

An algorithm for frequency estimation is characterized by two related parameters: the space[1] and the bounds on the error in es-

[1]We measure space in memory words, each consisting of a logarithmic number of bits.

Algorithm	Type	Space	Error bound
FREQUENT [13, 26, 23]	Counter	$O(1/\epsilon)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon F_1$
FREQUENT [6]	Counter	$O(1/\epsilon)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon F_1^{res(1)}$
LOSSYCOUNTING [24]	Counter	$O(1/\epsilon \log(\epsilon F_1))$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon F_1$
SPACESAVING [25]	Counter	$O(1/\epsilon)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon F_1$
Count-Min [12]	Sketch	$O((k/\epsilon) \cdot \log n)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon/k \cdot F_1^{res(k)}$
Count-Sketch [9]	Sketch	$O((k/\epsilon) \cdot \log n)$	$(f_i - \hat{f}_i)^2 \leq \epsilon/k \cdot F_2^{res(k)}$
This paper	Counter	$O(k/\epsilon)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon/k \cdot F_1^{res(k)}$

Table 1: Previously known bounds of frequency estimation algorithms.
F_1 is the sum of all frequencies; $F_1^{res(k)}$ is the sum of all but the top k frequencies; $F_2^{res(k)}$ is the sum of the squares of all but the top k frequencies; n is the size of the domain from which the stream elements are drawn.

timating the f_is. The error bounds are typically of the "additive" form, namely we have $|f_i - \hat{f}_i| \leq \epsilon B$, for a B (as in "bound") that is a function of the stream. The bound B is equal either to the size of the whole stream (equivalently, to the quantity F_1, where $F_p = \sum_i (f_i)^p$), or to the size of the *residual* tail of the stream, given by $F_1^{res(k)}$, the sum of the frequencies of all elements other than the k most frequent ones (heavy hitters). The residual guarantee is more desirable, since it is always at least as good as the F_1 bound. More strongly, since streams from real applications often obey a very *skewed* frequency distribution, with the heavy hitters constituting the bulk of the stream, a residual guarantee is *asymptotically* better. In particular, in the extreme case when there are only k distinct elements present in the stream, the residual error bound is zero, i.e. the frequency estimation is exact.

Algorithms for this problem have fallen into two main classes: (deterministic) "counter" algorithms and (randomized) "sketch" algorithms. Table 1 summarizes the space and error bounds of some of the main examples of such algorithms. As is evident from the table, the bounds for the counter and sketching algorithms are incomparable: counter algorithms use less space, but have worse error guarantees than sketching algorithms. In practice, however, the *actual* performance of counter-based algorithms has been observed to be appreciably better than of the sketch-based ones, given the same amount of space [10]. The reason for this disparity has not previously been well understood or explained. This has led users to apply very conservative bounds in order to provide the desired guarantees; it has also pushed users towards sketch algorithms in favor of counter algorithms since the latter are not perceived to offer the same types of guarantee as the former.

Our Contributions. In this paper we show that the good empirical performance of counter-based algorithms is not an accident: they actually *do* satisfy a much stronger error bound than previously thought. Specifically:

- We identify a general class of *Heavy-Tolerant Counter algorithms* (HTC), that contains the most popular FREQUENT and SPACESAVING algorithms. The class captures the essential properties of the algorithms and abstracts away from the specific mechanics of the procedures.
- We show that any HTC algorithm that has an ϵF_1 error guarantee in fact satisfies the stronger residual guarantee.

We conclude that FREQUENT and SPACESAVING offer the residual bound on error, while using less space than sketching algorithms. Moreover, counter algorithms have small constants of proportionality hidden in their asymptotic cost compared to the much larger logarithmic factors of sketch algorithms, making these space savings very considerable in practice. We also establish through a lower bound that the space usage of these algorithms is within a small constant factor of the space required by any counter algorithm that offers the residual bound on error.

The new bounds have several consequences beyond the immediate practical ramifications. First, we show that they provide better bounds for the *sparse recovery* problem, a streaming analog of Compressed Sensing [14, 7, 17, 28]. This problem is to find the best representation f^* of the frequency distribution, so that f^* has only k non-zero entries. Such a representation captures exact stream statistics for all but $\|f - f^*\|_1$ stream elements. We show that using a counter algorithm to produce the k largest estimated frequencies $\hat{f}_i$ yields a good solution to this problem. Formally, let S be the set of the k largest entries in $\hat{f}$, generated by a counter algorithm with $O(k/\epsilon)$ counters. Let f^* be an n-dimensional vector such that f_i^* is equal to $\hat{f}_i$ if $i \in S$ and $f_i^* = 0$ otherwise. Then we show that under the L_p norm, for any $p \geq 1$, we have

$$\|f - f^*\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p}$$

This is the best known result for this problem in a streaming setting; note that the error is always at least $(F_p^{res(k)})^{1/p}$. The best known sketching algorithms achieve this bound using $\Omega(k \log \frac{n}{k})$ space (see [2, 3, 22]); in contrast, our approach yields a space bound of $O(k)$. By extracting all m approximated values from a counter algorithm (as opposed to just top k), we are able to show another result. Specifically, by modifying the algorithms to ensure that they always provide an *underestimate* of the frequencies, we show that the resulting reconstruction has L_p error $(1+\epsilon)(\epsilon/k)^{1-1/p} F_1^{res(k)}$ for any $p \geq 1$.

As noted above, many common frequency distributions are naturally skewed. We show that if the frequencies follow a Zipfian distribution with parameter $\alpha > 1$, then the same tail guarantee follows using only $O(\epsilon^{-1/\alpha})$ space. Lastly, we also discuss extensions to the cases when streams can include arbitrary weights for each occurrence of an item; and when multiple streams are summarized and need to be merged together into a single summary. We show how the algorithms considered can be generalized to handle both of these situations.

1.1 Related Work

There is a large body of algorithms proposed in the literature for heavy hitters problems and their variants; see [10] for a survey. Most of them can be classified as either *counter-based* or *sketch-based*. The first counter algorithm is due to Misra and Gries [26], which we refer to as FREQUENT. Several subsequent works discussed efficient implementation and improved guarantees for this

algorithm [13, 6]. In particular, Bose *et al.* showed that it offers an $F_1^{res(1)}$ guarantee [6]. Our main result is to improve this to $F_1^{res(k)}$, for a broader class of algorithms.

A second counter algorithm is the LOSSYCOUNTING algorithm of Manku and Motwani. This has been shown to require $O(1/\epsilon)$ counters over randomly ordered streams to give an ϵF_1 guarantee, but there are adversarial order streams for which it requires $O(1/\epsilon \log \epsilon n)$ [24]. Our results hold over all possible stream orderings.

The most recent counter solution is the SPACESAVING algorithm due to Metwally *et al.* [25]. The algorithm is shown to offer an F_1 guarantee, and also analyzed in the presence of data with Zipfian frequency distribution. Here, we show an $F_1^{res(k)}$ bound, and demonstrate similar bounds for Zipfian data for a larger class of counter algorithms.

Sketch algorithms are based on linear projections of the frequency vector onto a smaller sketch vector, using compact hash functions to define the projection. Guarantees in terms of $F_1^{res(k)}$ or $F_2^{res(k)}$ follow by arguing that the items with the k largest frequencies are unlikely to (always) collide under the random choice of the hash functions, and so these items can effectively be "removed" from consideration. Because of this random element, sketches are analyzed probabilistically, and have a probability of failure that is bounded by $1/n^c$ for a constant c (n is the size of the domain from which the stream elements are drawn). The Count-Sketch requires $O((k/\epsilon) \log n)$ counters to give guarantees on the sum of squared errors in terms of $F_2^{res(k)}$ [9]; the Count-Min sketch uses $O((k/\epsilon) \log n)$ counters to give guarantees on the absolute error in terms of $F_1^{res(k)}$ [12]. These two guarantees are incomparable in general, varying based on the distribution of frequencies. A key distinction of sketch algorithms is that they allow both positive and negative updates (where negative updates can correspond to deletions, in a transactional setting, or simply arbitrary signal values, in a signal processing environment). This, along with the fact that they are linear transforms, means that they can be used to solve problems such as designing measurements for compressed sensing systems [17, 7]. So, although our results show that counter algorithms are strictly preferable to sketches when both are applicable, there are problems that are solved by sketches that cannot be solved using counter algorithms.

We summarize the main properties of these algorithms, along with the correspond results based on our analysis, in Table 1.

2. PRELIMINARIES

We introduce the notation used throughout this paper. The algorithms maintain at most m counters which correspond to a "frequent" set of elements occurring in the input stream. The input stream contains elements, which we assume to be integers between 1 and n. We denote a stream of size N by $u_1, u_2, \ldots u_N$. We use $u_{x \ldots y}$ as a shorthand for the partial stream $u_x, u_{x+1}, \ldots, u_y$.

We denote frequencies of elements by an n-dimensional vector f. For ease of notation, we assume without loss of generality that elements are indexed in order of decreasing frequency, so that $f_1 \geq f_2 \geq \ldots \geq f_n$. When the stream is not understood from context, we specify it explicitly, e.g. $f(u_{x \ldots y})$ is the frequency vector for the partial stream $u_{x \ldots y}$. We denote the sum of the frequencies by F_1; we denote the sum of frequencies except the largest ones by $F_1^{res(k)}$, and we generalize the definition to sums of powers of the frequencies:

$$F_p^{res(k)} = \sum_{i=k+1}^{n} f_i^p, \qquad F_p = F_p^{res(0)}$$

The algorithms considered in this paper can be thought of as adhering to the following form. The state of an algorithm is represented by an n-dimensional vector of counters c. The vector c has at most m non-zero elements. We denote the "frequent" set by $T = \{i \mid c_i \neq 0\}$, since only this set needs to be explicitly stored. The counter value of an element is an approximation for its frequency; the error vector of the approximation is denoted by δ, with $\delta_i = |f_i - c_i|$.

We demonstrate our results with reference to two known counter algorithms: FREQUENT and SPACESAVING. Although similar, the two algorithms differ in the analysis and their behavior in practice. Both maintain their frequent set T, and process a stream of updates. Given a new item i in the stream which is stored in T, both simply increase the corresponding counter c_i; or, if $i \notin T$ and $|T| < m$, then i is stored with a count of 1. The algorithms differ when an unstored item is seen and $|T| = m$: FREQUENT decrements all stored counters by 1, and (implicitly) throws out any counters with zero count; SPACESAVING finds an item j with smallest non-zero count c_j and assigns $c_i \leftarrow c_j + 1$, followed by $c_j \leftarrow 0$, so in effect i replaces j in T. Pseudocode for these algorithms is presented in Figure 1

These algorithms are known to provide a "heavy hitter" guarantee on the approximation errors of the counters:

Definition 1. An m-counter algorithm provides a *heavy hitter guarantee with constant* $A > 0$ if, for any stream,

$$\delta_i \leq \left\lfloor A \frac{F_1}{m} \right\rfloor \qquad \forall i$$

More precisely, they both provide this guarantee with constant $A = 1$. Our result is that they also satisfy the following stronger guarantee:

Definition 2. An m-counter algorithm provides a *k-tail guarantee with constants* (A, B), *with* $A, B > 0$ if for any stream

$$\delta_i \leq \left\lfloor A \frac{F_1^{res(k)}}{m - Bk} \right\rfloor \qquad \forall i$$

Note that the heavy hitter guarantee is equivalent to the 0-tail guarantee. Our general proof (which can be applied to a broad class of algorithms) yields a k-tail guarantee with constants $A = 1$, $B = 2$ for both algorithms (for any $k \leq m/2$). However, by considering particular features of FREQUENT and SPACESAVING, we prove a k-tail guarantee with constants $A = B = 1$ for any $k < m$ following appropriate analysis (see appendices B, C).

The lower bound proved in appendix A establishes that any counter algorithm that provides an error bound of $\frac{F_1^{res(k)}}{m-k}$ must use at least $(m - k)/2$ counters; thus the number of counters FREQUENT and SPACESAVING use is within a small factor (3 for $k \leq m/3$) of the optimal.

3. RESIDUAL ERROR BOUND

In this section we state and prove our main result on the error bound for a class of heavy-tolerant counter algorithms. We begin by formally defining this class.

Definition 3. A value i is *x-prefix guaranteed* for the stream $u_{1 \ldots s}$ if after the first $x < s$ elements of the stream have been processed, i will stay in T even if some elements are removed from the remaining stream (including occurrences of i). Formally, the value i is *x-prefix guaranteed* if $0 \leq x < s$ and $c_i(u_{1 \ldots x} v_{1 \ldots t}) > 0$ for all subsequences $v_{1 \ldots t}$ of $u_{(x+1) \ldots s}$, $0 \leq t \leq s - x$.

Algorithm 1: FREQUENT(m)

$T \leftarrow \emptyset$;
foreach i **do**
  if $i \in T$ **then**
    $c_i \leftarrow c_i + 1$;
  else if $|T| < m$ **then**
    $T \leftarrow T \cup \{i\}$;
    $c_i \leftarrow 1$;
  else forall $j \in T$ **do**
    $c_j \leftarrow c_j - 1$;
    if $c_j = 0$ **then**
      $T \leftarrow T \backslash \{j\}$;

Algorithm 2: SPACESAVING(m)

$T \leftarrow \emptyset$;
foreach i **do**
  if $i \in T$ **then**
    $c_i \leftarrow c_i + 1$;
  else if $|T| < m$ **then**
    $T \leftarrow T \cup \{i\}$;
    $c_i \leftarrow 1$;
  else
    $j \leftarrow \arg\min_{j \in T} c_j$;
    $c_i \leftarrow c_j + 1$;
    $T \leftarrow T \cup \{i\} \backslash \{j\}$;

Figure 1: Pseudocode for FREQUENT **and** SPACESAVING **algorithms**

Note that if i is x-prefix guaranteed, then i is also y-prefix guaranteed for all $y > x$.

Definition 4. A counter algorithm is *heavy-tolerant* if extra occurrences of guaranteed elements do not increase the estimation error. Formally, an algorithm is *heavy-tolerant* if for any stream $u_{1\ldots s}$, given any x, $1 \leq x < s$, for which element $i = u_x$ is $(x-1)$-prefix guaranteed, it holds that

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots(x-1)}u_{(x+1)\ldots s}) \qquad \forall j$$

THEOREM 1. *Algorithms* FREQUENT *and* SPACESAVING *are heavy-tolerant.*

THEOREM 2. *If a heavy-tolerant algorithm provides a heavy hitter guarantee with constant A, it also provides a k-tail guarantee with constants $(A, 2A)$, for any k, $1 \leq k < m/2A$.*

3.1 Proof of Heavy Tolerance

Intuitively, this is true because occurrences of an element already in the frequent set only affect the counter value of that element; and, as long as the element never leaves the frequent set, the value of its counter does not affect the algorithm's other choices.

PROOF OF THEOREM 1. Denote $v_{1\ldots t} = u_{(x+1)\ldots(x+t)}$, with $t \leq s - x$. We prove by induction on t that for both algorithms

$$c(u_{1\ldots x}v_{1\ldots t}) = c(u_{1\ldots(x-1)}v_{1\ldots t}) + e_i$$

where e_i is the i-th row of I_n, the $n \times n$ identity matrix; this implies that

$$\delta(u_{1\ldots x}v_{1\ldots t}) = \delta(u_{1\ldots(x-1)}v_{1\ldots t})$$

Base case at $t = 0$: By the hypothesis: $c_i(u_{1\ldots(x-1)}) \neq 0$, hence when element $u_x = i$ arrives after processing $u_{1\ldots x}$, both FREQUENT and SPACESAVING just increase i's counter:

$$c(u_{1\ldots x}) = c(u_{1\ldots(x-1)}) + e_i$$

Induction step for $t > 0$: We are given that

$$c(u_{1\ldots x}v_{1\ldots(x-1)}) = c(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) + e_i$$

Note that since i is $(x-1)$-prefix guaranteed, these vectors have the same support.

Case 1: $c_{v_t}(u_{1\ldots x}v_{1\ldots(t-1)}) > 0$. Hence $c_{v_t}(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) > 0$. For both streams, v_t's counter just gets incremented and thus

$$\begin{aligned} c(u_{1\ldots x}v_{1\ldots t}) &= c(u_{1\ldots x}v_{1\ldots(t-1)}) + e_{v_t} \\ &= c(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) + e_{v_t} + e_i \\ &= c(u_{1\ldots(x-1)}v_{1\ldots t}) + e_i \end{aligned}$$

Case 2: $c_{v_t}(u_{1\ldots x}v_{1\ldots(t-1)}) = 0$. Note that $v_t \neq i$ since i is x-prefix guaranteed and $c_{v_t}(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) = 0$. By the induction hypothesis, both counter vectors have the same support (set of non-zero entries). If the support is less than m, then the algorithm adds e_{v_k} to the counters, and the analysis follows Case 1 above. Otherwise, the two algorithms differ:

- FREQUENT algorithm: In this case all non-zero counters will be decremented. Since both counter vectors have the same support, they will be decremented by the same m-sparse binary vector $\gamma = \chi(T) = \sum_{j:c_j \neq 0} e_j$.
- SPACESAVING algorithm: The minimum non-zero counter is set to zero. To avoid ambiguity, we specify that SPACESAVING will pick the counter c_j with the smallest identifier j if there are multiple counters with equal smallest non-zero value. Let

  $$j = \operatorname*{argmin}_{j \in T(u_{1\ldots x}v_{1\ldots(t-1)})} c_j(u_{1\ldots x}v_{1\ldots(t-1)})$$

  and

  $$j' = \operatorname*{argmin}_{j' \in T(u_{1\ldots(x-1)}v_{1\ldots(t-1)})} c_{j'}(u_{1\ldots(x-1)}v_{1\ldots(t-1)})$$

  Since i is x-prefix guaranteed, its counter can never become zero, hence $j \neq i, j' \neq i$. Since

  $$c_{i'}(u_{1\ldots x}v_{1\ldots(t-1)}) = c_{i'}(u_{1\ldots(x-1)}v_{1\ldots(t-1)})$$

  for all $i' \neq i$, it follows that $j = j'$ and

  $$c_j(u_{1\ldots x}v_{1\ldots(t-1)}) = c_{j'}(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) = M.$$

  Hence both streams result in updating the counters by subtracting the same difference vector $\gamma = Me_j - (M+1)e_{v_t}$

So each algorithm computes some difference vector γ irrespective of which stream it is applied to, and updates the counters:

$$\begin{aligned} c(u_{1\ldots x}v_{1\ldots t}) &= c(u_{1\ldots x}v_{1\ldots(t-1)}) - \gamma \\ &= c(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) + e_i - \gamma \\ &= c(u_{1\ldots(x-1)}v_{1\ldots t}) + e_i \end{aligned}$$

□

3.2 Proof of k-tail guarantee

Let $\texttt{Remove}(u_{1\ldots s}, i)$ be the subsequence of $u_{1\ldots s}$ with all occurrences of value i removed, i.e.

$$\texttt{Remove}(u_{1\ldots s}, i) = \begin{cases} \text{empty sequence} & \text{if } s = 0 \\ (u_1, \texttt{Remove}(u_{2\ldots s}, i)) & \text{if } u_1 \neq i \\ \texttt{Remove}(u_{2\ldots s}, i) & \text{if } u_1 = i \end{cases}$$

LEMMA 3. *If i is x-prefix guaranteed and the algorithm is heavy-tolerant, then*

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots x} v_{1\ldots t}) \qquad \forall j$$

where $v_{1\ldots t} = \texttt{Remove}(u_{(x+1)\ldots s}, i)$, with $0 \leq t \leq s - x$.

PROOF. Let $x_1, x_2, \ldots, x_q$ be the positions of occurrences of i in $u_{(x+1)\ldots s}$, with $x < x_1 < x_2 < \ldots < x_q$. We apply the heavy-tolerant definition for each occurrence; for all j:

$$\begin{aligned} \delta_j(u_{1\ldots s}) &\leq \delta_j(u_{1\ldots(x_1-1)} u_{(x_1+1)\ldots s}) \\ &\leq \delta_j(u_{1\ldots(x_1-1)} u_{(x_1+1)\ldots(x_2-1)} u_{(x_2+1)\ldots s}) \\ &\leq \ldots \\ &\leq \delta_j(u_{1\ldots x} v_{1\ldots t}) \end{aligned}$$

Note in particular that $\delta_i(u_{1\ldots p})$, the error in estimating the frequency of i in the original stream, is identical to $\delta_i(u_{1\ldots x} v_{1\ldots q})$, the error of i on the derived stream, since i is x-prefix guaranteed. □

Definition 5. An *error bound* for an algorithm is a function $\Delta : \mathbb{N}^n \to \mathbb{R}^+$ such that for any stream $u_{1\ldots s}$

$$\delta_i(u_{1\ldots s}) \leq \lfloor \Delta(f(u_{1\ldots s})) \rfloor \qquad \forall i$$

In addition, Δ must be "increasing" in the sense that for any two frequency vectors f' and f'' such that $f'_i \leq f''_i$ for all i, it holds that $\Delta(f') \leq \Delta(f'')$.

LEMMA 4. *Let Δ be an error bound for a heavy-tolerant algorithm that provides a heavy hitter guarantee with constant A. Then the following function is also an error bound for the algorithm, for any k, $1 \leq k < m/A$:*

$$\Delta'(f) = A \frac{k\Delta(f) + k + F_1^{res(k)}}{m}$$

PROOF. Let $u_{1\ldots s}$ be any stream. Let $D = 1 + \lfloor \Delta(f(u_{1\ldots s})) \rfloor$. We assume without loss of generality that the elements are indexed in order of increasing frequency.

Let $k' = \max\{i \mid 1 \leq i \leq k \text{ and } f_i(u_{1\ldots s}) > D\}$.

For each $i \leq k'$ let x_i be the position of the D-th occurrence of i in the stream. We claim that any $i \leq k'$ is x_i-prefix guaranteed: let $v_{1\ldots t}$ be any subsequence of $u_{(x_i+1)\ldots s}$; it holds for all j that

$$\delta_j(u_{1\ldots x_i} v_{1\ldots t}) \leq \lfloor \Delta(f(u_{1\ldots x_i} v_{1\ldots t})) \rfloor < D$$

$$\begin{aligned} \text{and so } c_j(u_{1\ldots x_i} v_{1\ldots t}) &\geq f_j(u_{1\ldots x_i} v_{1\ldots t}) - \delta_j(u_{1\ldots x_i} v_{1\ldots t}) \\ &> D - D = 0. \end{aligned}$$

Let $i_1, i_2, \ldots i_{k'}$ be the permutation of $1 \ldots k'$ so that $x_{i_1} > x_{i_2} > \ldots > x_{i_{k'}}$. We can apply Lemma 3 for i_1 which is x_{i_1}-prefix guaranteed; for all j

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots x_{i_1}} v_{1\ldots s_v})$$

where $v_{1\ldots s_v} = \texttt{Remove}(u_{(x_{i_1}+1)\ldots s}, i_1)$.

Since $x_2 < x_1$, i_2 is x_2-prefix guaranteed for the new stream $u_{1\ldots x_{i_1}} v_{1\ldots s_v}$ and we apply Lemma 3 again:

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots x_{i_1}} v_{1\ldots s_v}) \leq \delta_j(u_{1\ldots x_{i_2}} w_{1\ldots s_w}) \qquad \forall j$$

where $w_{1\ldots s_w} = \texttt{Remove}(u_{(x_{i_2}+1)\ldots x_{i_1}} v_{1\ldots s_v}, i_2)$. Since the x_{i_j} values are decreasing, we can continue this argument for $i = 3, 4, \ldots, k'$. We obtain the following inequality for the final stream $z_{1\ldots s_z}$

$$\delta_j(u_{1\ldots s}) \leq \delta_j(z_{1\ldots s_z}) \qquad \forall j$$

where $z_{1\ldots s_z}$ is the stream $u_{1\ldots s}$ with all "extra" occurrences of elements 1 to k' removed ("extra" means after the first D occurrences). Thus

$$\|f(z_{1\ldots s_z})\|_1 = k'D + \sum_{i=k'+1}^{n} f_i(u_{1\ldots s})$$

Either $k' = k$, or $k' < k$ and $f_i(u_{1\ldots s}) \leq D$ for all $k' < i \leq k$; in both cases we can replace k' with k:

$$\|f(z_{1\ldots s_z})\|_1 \leq kD + \sum_{i=k+1}^{n} f_i(u_{1\ldots s})$$

We now apply the heavy hitter guarantee for this stream; for all j:

$$\begin{aligned} \delta_j(u_{1\ldots s}) &\leq \delta_j(z_{1\ldots s_z}) \\ &\leq \left\lfloor A \frac{kD + \sum_{i=k+1}^{n} f_i(u_{1\ldots s})}{m} \right\rfloor \\ &\leq \left\lfloor A \frac{k\Delta(u_{1\ldots s}) + k + F_1^{res(k)}}{m} \right\rfloor \end{aligned}$$

□

We can now prove theorem 2.

PROOF OF THEOREM 2. We start with the initial error bound given by the heavy hitter guarantee $\Delta(f) = A\frac{\|f\|_1}{m}$ and apply Lemma 4 to obtain another error bound Δ'. We can continue iteratively applying Lemma 4 in this way. Either we will eventually obtain a new bound which is worse than the previous one, in which case this process halts with the previous error bound; or else we can analyze the error bound obtained in the limit (in the spirit of [6]). In both cases, the following holds for the best error bound Δ:

$$\Delta(f) \leq A \frac{k\Delta(f) + k + F_1^{res(k)}}{m}$$

$$\text{and so } \Delta(f) \leq A \frac{k + F_1^{res(k)}}{m - Ak} .$$

We have shown that for any stream $u_{1\ldots p}$,

$$\delta_i(u_{1\ldots p}) \leq \left\lfloor A \frac{k + F_1^{res(k)}}{m - Ak} \right\rfloor \qquad \forall i$$

We show that this implies the guarantee

$$\delta_i(u_{1\ldots p}) \leq \left\lfloor A \frac{F_1^{res(k)}}{m - 2Ak} \right\rfloor \qquad \forall i$$

Case 1: $AF_1^{res(k)} < m - 2Ak$. In this case both guarantees are identical: all errors are 0.

Case 2: $AF_1^{res(k)} \geq m - 2Ak$:

$$\begin{aligned} A^2 k F_1^{res(k)} &\geq Ak(m - 2Ak) \\ A(m - Ak) F_1^{res(k)} &\geq A(m - 2Ak)\left(k + F_1^{res(k)}\right) \\ A \frac{F_1^{res(k)}}{m - 2Ak} &\geq A \frac{k + F_1^{res(k)}}{m - Ak} \end{aligned}$$

□

4. SPARSE RECOVERIES

The k-sparse recovery problem is to find a representation f' so that f' has only k non-zero entries ("k-sparse"), and the L_p norm $\|f - f'\|_p = (\sum_{i=1}^{n} |f_i - f'_i|^p)^{1/p}$ is minimized. A natural approach is to build f' from the heavy hitters of f, and indeed we show that this method gives strong guarantees for frequencies from heavy tolerant counter algorithms.

4.1 k-sparse recovery

To get a k-sparse recovery, we run counter algorithm that provides a k-tail guarantee with m counters and create f' using the k largest counters. These are not necessarily the k most frequent elements (with indices 1 to k in our notation), but we show that they must be "close enough".

THEOREM 5. *If we run a counter algorithm which provides a k-tail guarantee with constants (A, B) using $m = k(\frac{3A}{\varepsilon} + B)$ counters and retain the top k counter values into the k-sparse vector f', then for any $p \geq 1$:*

$$\|f - f'\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p}$$

PROOF. Let $K = \{1, \ldots, k\}$ be the set of the k most frequent elements. Let S be the set of elements with the k largest counters. Let $R = \{1, \ldots, n\} \setminus (S \cup K)$ be the set of all other remaining elements. Let $k' = |K \setminus S| = |S \setminus K|$.

Let $x_1 \ldots x_{k'}$ be the k' elements in $S \setminus K$, with $c_{x_1} \geq c_{x_2} \geq \ldots \geq c_{x_{k'}}$. Let $y_1 \ldots y_{k'}$ be the k' elements in $K \setminus S$, with $c_{y_1} \geq c_{y_2} \geq \ldots \geq c_{y_{k'}}$. Notice that $c_{x_i} \geq c_{y_i}$ for any i: c_{y_i} is the i^{th} largest counter in $K \setminus S$, whereas c_{x_i} is the i^{th} largest counter in $(K \cup S) \setminus (S \cap K)$, a superset of $K \setminus S$. Let Δ be an upper bound on the counter errors δ. Then for any i

$$f_{y_i} - \Delta \leq c_{y_i} \leq c_{x_i} \leq f_{x_i} + \Delta \tag{1}$$

Hence $f_{y_i} \leq f_{x_i} + 2\Delta$. Let f' be the recovered frequency vector ($f'_{x_i} = c_{x_i}$ and zero everywhere else). For any $p \geq 1$, and using the triangle inequality $\|a + b\|_p \leq \|a\|_p + \|b\|_p$ on the vector f_i restricted to $i \in R \cup S$ and the vector equal to the constant 2Δ restricted to $i \in S \setminus K$:

$$\begin{aligned}
\|f - f'\|_p &= \left(\sum_{i \in S} (c_i - f_i)^p + \sum_{i \in R \cup K \setminus S} (f_i)^p\right)^{1/p} \\
&\leq \left(\sum_{i=1}^{k} \Delta^p + \sum_{i \in K \setminus S} (f_i)^p + \sum_{i \in R} (f_i)^p\right)^{1/p} \\
&\leq k^{1/p}\Delta + \left(\sum_{i=1}^{k'} (f_{y_i})^p + \sum_{i \in R} (f_i)^p\right)^{1/p} \\
&\leq k^{1/p}\Delta + \left(\sum_{i=1}^{k'} (f_{x_i} + 2\Delta)^p + \sum_{i \in R} (f_i)^p\right)^{1/p} \\
&\leq 3k^{1/p}\Delta + \left(\sum_{i \in R \cup S \setminus K} (f_i)^p\right)^{1/p} \\
&\leq 3k^{1/p}\Delta + (F_p^{res(k)})^{1/p}
\end{aligned}$$

If an algorithm has the tail guarantee with constants (A, B), by using $m = k(\frac{3A}{\varepsilon} + B)$ counters we get

$$\|f - f'\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p} \tag{2}$$

□

Note that $(F_p^{res(k)})^{1/p}$ is the smallest possible L_p error of any k-sparse recovery of f. Also, if the algorithm provides one-sided error on the estimated frequencies (as is the case for FREQUENT and SPACESAVING), it is sufficient to use $m = k(\frac{2A}{\varepsilon} + B)$ counters, since now $f_{y_i} \leq f_{x_i} + \Delta$.

Estimating $F_1^{res(k)}$. Since our algorithms give guarantees in terms of $F_1^{res(k)}$, a natural question is to estimate the value of this quantity.

THEOREM 6. *If we run a counter algorithm which provides a k-tail guarantee with constants (A, B) using $(Bk + \frac{Ak}{\varepsilon})$ counters and retain the largest k counter values as the k-sparse vector f', then:*

$$F_1^{res(k)}(1 - \varepsilon) \leq F_1 - \|f'\|_1 \leq F_1^{res(k)}(1 + \varepsilon)$$

PROOF. To show this result, we rely on the definitions and properties of sets S and K from the proof of Theorem 5. By construction of sets S and K, $f_{x_i} \leq f_{y_i}$ for any i. Using equation (1) it follows that

$$f_{y_i} - \Delta \leq c_{x_i} \leq f_{y_i} + \Delta$$

So the norm of f' must be close to the norm of the best k-sparse representative of f, i.e. $(F_1 - F_1^{res(k)})$. Summing over each of the k counters yields

$$\begin{aligned}
F_1 - F_1^{res(k)} - k\Delta \leq \quad & \|f'\|_1 \quad \leq F_1 - F_1^{res(k)} + k\Delta \\
F_1^{res(k)} - k\Delta \leq \quad F_1 - \|f'\|_1 \quad & \leq F_1^{res(k)} + k\Delta
\end{aligned}$$

The result follows when setting $m = k(\frac{Ak}{\varepsilon} + B))$ so the upper bound ensures $\Delta \leq \frac{\varepsilon}{k} F_1^{res(k)}$. □

4.2 m-sparse recovery

When the counter algorithm uses m counters, it stores approximate values for m elements. It seems intuitive that by using all m of these counter values, the recovery should be even better. This turns out not to be true in general. Instead, we show that it is possible to derive a better result given an algorithm which always *underestimates* the frequencies ($c_i \leq f_i$). For example, this is true in the case of FREQUENT.

As described so far, SPACESAVING always overestimates, but can be modified to underestimate the frequencies. In particular, the algorithm has the property that error is bounded by the smallest counter value, i.e. $\Delta = \min\{c_j | c_j \neq 0\}$. So setting $c'_i = \max\{0, c_i - \Delta\}$ ensures that $c'_i \leq f_i$. Because $f_i + \Delta \geq c_i \geq f_i$, $f_i - c'_i \leq \Delta$ and thus c' satisfies the same k-tail bounds with $A = B = 1$ (as per appendix C). Note that in practice, slightly improved per-item guarantees follow by storing ϵ_i for each non-zero counter c_i as the value of Δ when i last entered the frequent set, and using $c_i - \epsilon_i$ as the estimated value (as described in [25]).

THEOREM 7. *If we run an underestimating counter algorithm which provides a k-tail guarantee with constants (A, B) using $(Bk + \frac{Ak}{\varepsilon})$ counters and retain the counter values into the m-sparse vector f', then for any $p \geq 1$:*

$$\|f - f'\|_p \leq (1 + \varepsilon) \left(\frac{\varepsilon}{k}\right)^{1-1/p} F_1^{res(k)}$$

PROOF. Set $m = k(\frac{A}{\varepsilon} + B)$ in Definition 2 to obtain

$$\begin{aligned}\|f - f'\|_p &= \left(\sum_{i=1}^{k}(f_i - c_i)^p + \sum_{i=k+1}^{n}(f_i - c_i)^p\right)^{1/p} \\ &\leq \left(k\frac{\varepsilon^p}{k^p}(F_1^{res(k)})^p + \right. \\ &\quad \left. + \sum_{i=k+1}^{n}(f_i - c_i)\frac{\varepsilon^{p-1}}{k^{p-1}}(F_1^{res(k)})^{p-1}\right)^{1/p} \\ &\leq \left(\frac{\varepsilon^p}{k^{p-1}}(F_1^{res(k)})^p + \frac{\varepsilon^{p-1}}{k^{p-1}}(F_1^{res(k)})^p\right)^{1/p} \\ &\leq (1+\varepsilon)\left(\frac{\varepsilon}{k}\right)^{1-1/p} F_1^{res(k)}\end{aligned}$$

□

5. ZIPFIAN DISTRIBUTIONS

Realistic data can often be approximated with a Zipfian [30] distribution; a stream of length $F_1 = N$, with n distinct elements, distributed (exactly) according to the Zipfian distribution with parameter α has frequencies

$$f_i = N\frac{1}{i^\alpha \zeta(\alpha)} \quad \text{where} \quad \zeta(\alpha) = \sum_{i=1}^{n}\frac{1}{i^\alpha}$$

The value $\zeta(\alpha)$ converges to a small constant when $\alpha > 1$. Although data rarely obeys this distribution exactly, our first result requires only that the "tail" of the distribution can be bounded by a (small constant multiple of) a Zipfian distribution. Note that this requires that the frequencies follow this distribution, but the order of items in the stream can be arbitrary.

THEOREM 8. *Given Zipfian data with parameter $\alpha \geq 1$, if a counter algorithm that provides a k-tail guarantee with constants (A, B) for $k = \left(\frac{1}{\varepsilon}\right)^{1/\alpha}$ is used with $m = (A+B)\left(\frac{1}{\varepsilon}\right)^{1/\alpha}$ counters, the counter errors are at most εF_1.*

PROOF. The k-tail guarantee with constants (A, B) means

$$\Delta = A\frac{F_1^{res(k)}}{m - Bk} \leq A\frac{N}{\zeta(\alpha)}\frac{\sum_{i=k+1}^{n} i^{-\alpha}}{m - Bk}$$

Then

$$\sum_{i=k+1}^{n}\frac{1}{i^\alpha} \leq \int_k^n \frac{1}{x^\alpha}dx = \frac{1}{k^{\alpha-1}}\int_1^{n/k}\frac{1}{x^\alpha}dx \leq \frac{\zeta(\alpha)}{k^{\alpha-1}}$$

$$\Delta \leq A\frac{\zeta(\alpha)}{k^{\alpha-1}}\frac{N}{\zeta(\alpha)(m - Bk)} = \frac{N}{k^\alpha}A\frac{k}{m - Bk}$$

by setting $k = \left(\frac{1}{\varepsilon}\right)^{1/\alpha}$, $m = (A+B)k$,

$$\Delta \leq \frac{N}{k^\alpha} = \varepsilon N$$

□

A similar result is proved for SPACESAVING in [25] under the stronger assumption that the frequencies are exactly as defined by the Zipfian distribution.

5.1 Top-k

In this section we analyze the algorithms in the context of the problem of finding top k elements, when the input is Zipf distributed.

THEOREM 9. *Assuming Zipfian data with parameter $\alpha > 1$, a counter algorithm that provides a k'-tail guarantee for $k' = \Theta\left(k\left(\frac{k}{\alpha}\right)^{1/\alpha}\right)$ can retrieve the top k elements in correct order using $O\left(k\left(\frac{k}{\alpha}\right)^{1/\alpha}\right)$ counters. For Zipfian data with parameter $\alpha = 1$, an algorithm with k'-tail guarantee for $k' = \Theta(k^2 \ln n)$ can retrieve the top k elements in correct order using $O(k^2 \ln n)$ counters.*

PROOF. To get the top k elements in the correct order we need

$$\Delta < \frac{f_k - f_{k+1}}{2}$$

$$\begin{aligned}f_k - f_{k+1} &= \frac{N}{\zeta(\alpha)}\left(\frac{1}{k^\alpha} - \frac{1}{(k+1)^\alpha}\right) \\ &= \frac{N}{\zeta(\alpha)}\frac{(k+1)^\alpha - k^\alpha}{(k+1)^\alpha k^\alpha} \\ &< \frac{N}{\zeta(\alpha)}\frac{\alpha k^{\alpha-1}}{(k+1)^\alpha k^\alpha} = \frac{N}{\zeta(\alpha)}\frac{\alpha}{(k+1)^\alpha k}\end{aligned}$$

Thus we need error rate

$$\varepsilon = \frac{\alpha}{2\zeta(\alpha)(k+1)^\alpha k} = \begin{cases}\Theta(\alpha/k^{1+\alpha}) & \text{for } \alpha > 1 \\ \Theta(1/(k^2 \ln n)) & \text{for } \alpha = 1\end{cases}$$

The result then follows from Theorem 8. □

6. EXTENSIONS

6.1 Real-Valued Update Streams

So far, we have considered a model of streams where each stream token indicates an arrival of an item with (implicit) unit weight. More generally, streams often include a weight for each arrival: a size in bytes or round-trip time in seconds for Internet packets; a unit price for transactional data, and so on. When these weights are large, or not necessarily integral, it is still desirable to solve heavy hitters and related problems on such streams.

In this section, we make the observation that the two counter algorithms FREQUENT and SPACESAVING naturally extend to streams in which each update includes a positive real valued weight to apply to the given item. That is, the stream consists of tuples u_i, Each u_i is a tuple (a_i, b_i) representing b_i occurrences of element a_i where $b_i \in \mathbb{R}^+$ is a positive real value.

We outline how to extend the two algorithms to correctly process such streams. For SPACESAVING, observe that when processing each new item a_i, the algorithm identifies a counter corresponding to a_i and increments it by 1. We simply change this to incrementing the appropriate counter by b_i to generate an algorithm we denote SPACESAVING$\mathbb{R}$. It is straightforward to modify the analysis of [25] to demonstrate that SPACESAVING$\mathbb{R}$ achieves the basic Heavy Hitters guarantee (Definition 1). This generalizes SPACESAVING, since when every b_i is 1, then the two algorithms behave identically.

Defining FREQUENT$\mathbb{R}$ is a little more complex. If the new item $a_i \in T$, then we can simply increases a_i's counter by b_i; and if there are fewer than $m-1$ counters then one can be allocated to a_i and set to b_i. But if a_i is not stored, then the next step depends on the size of $c_{\min}$, the smallest counter value stored in T.

If $b_i \leq c_{\min}$, then all stored counters are reduced by b_i. Otherwise, all counters are reduced by $c_{\min}$, and some counter with zero count (there must be at least one now) is assigned to a_i and given count $b_i - c_{\min}$. Following this, items with zero count are removed from T. Then FREQUENT$\mathbb{R}$ achieves the basic Heavy Hitter guarantee by observing that every subtraction of counter values for a given item coincides with the same subtraction to $m-1$ others, and all counter increments correspond to some b_i of a particular item. Therefore, the error in the count of any item is at most F_1/m.

We comment that a similar analysis to that provided in Section 3 applies, to demonstrate that these new counter algorithms give a tail guarantee. The main technical challenge is generalizing the definitions of x-prefix guaranteed and heavy tolerant algorithms in the presence of arbitrary real updates. We omit the detailed analysis from this presentation, and instead we state in summary:

THEOREM 10. FREQUENT$\mathbb{R}$ *and* SPACESAVING$\mathbb{R}$ *both provide k-tail guarantees with $A = B = 1$ over real-valued non-negative update streams.*

6.2 Merging Multiple Summaries

A consequence of sparse recovery is the fact that multiple summaries of separate streams can be merged together to create a summary of the union of the streams. More formally, consider ℓ streams, defining frequency distributions $f^{(1)} \ldots f^{(\ell)}$ respectively. Given a summary of each stream produced by (the same) algorithm with m counters, the aim is to construct an accurate summary of $f = \sum_{j=1}^{\ell} f^{(j)}$.

THEOREM 11. *Given summaries of each $f^{(j)}$ produced by a counter algorithm that provides a k-tail guarantee with constants (A, B), a summary of f can be obtained with a k-tail guarantee with constants $(3A, B + A)$.*

PROOF. We construct a summary by first building a k-sparse vector $f'^{(j)}$ from the summary of $f^{(j)}$, with the guarantee of equation (2). By generating a stream corresponding to this vector for each stream, and feeding this into the counter algorithm, we obtain a summary of the distribution $f' = \sum_{j=1}^{\ell} f'^{(j)}$. Now observe that from this we have an estimated frequency for any item i as c_i so that

$$|c_i - f_i| \leq \Delta = \Delta_{f'} + \sum_{j=1}^{\ell} \Delta_j$$

where each Δ_j is the error from summarizing $f^{(j)}$ by $f'^{(j)}$, while $\Delta_{f'}$ is the error from summarizing f'. For the analysis, we require the following bound:

LEMMA 12. *For any n-dimensional vectors x and y,*

$$|F_1^{res(k)}(x) - F_1^{res(k)}(y)| \leq \|x - y\|_1$$

PROOF. Let X denote the set of k largest entries of x, and Y the set of k largest entries of y. Let $\pi(i)$ determine any bijection from $i \in Y \backslash X$ to $\pi(i) \in X \backslash Y$. Then

$$\begin{aligned} F_1^{res(k)}(x) - F_1^{res(k)}(y) &= \sum_{i \notin X} x_i - \sum_{i \notin Y} y_i \\ &\leq \sum_{i \in Y \backslash X} x_{\pi(i)} - \sum_{i \in X \backslash Y} y_i + \sum_{i \notin (X \cup Y)} |x_i - y_i| \\ &= \sum_{i \notin Y} |x_i - y_i| \leq \sum_i |x_i - y_i| \leq \|x - y\|_1 \end{aligned}$$

Interchanging the roles of x and y gives the final result. □

This lets us place an upper bound on the first component of the error:

$$\begin{aligned} \Delta_{f'} &\leq \frac{A}{m - Bk} F_1^{res(k)}(f') \\ &\leq \frac{A}{m - Bk} (F_1^{res(k)}(f) + \|f - f'\|_1) \end{aligned}$$

where, by the triangle inequality and the proof of Theorem 5,

$$\begin{aligned} \|f - f'\|_1 &\leq \sum_{j=1}^{\ell} \|f^{(j)} - f'^{(j)}\|_1 \\ &\leq \sum_{j=1}^{\ell} (3k\Delta_j + F_1^{res(k)}(f^{(j)})) \end{aligned}$$

Since $\Delta_j \leq A F_1^{res(k)}(f^{(j)})/(m - Bk)$, the total error obeys

$$\Delta \leq \frac{A}{m - Bk} \left(F_1^{res(k)}(f) + \sum_{j=1}^{\ell} (3k\Delta_j + 2F_1^{res(k)}(f^{(j)}) \right)$$

We observe that

$$\sum_{j=1}^{\ell} F_1^{res(k)}(f^{(j)}) \leq F_1^{res(k)} \left(\sum_{j=1}^{\ell} f^{(j)} \right) = F_1^{res(k)}(f)$$

since $\sum_{j=1}^{\ell} F_1^{res(k)}(f^{(j)}) \leq \sum_{j=1}^{\ell} \sum_{i \notin T} f^{(j)}$ for any T such that $|T| = k$. So

$$\begin{aligned} \Delta &\leq \frac{A}{m - Bk} \left(3F_1^{res(k)}(f) + 3k \frac{A}{m - Bk} (F_1^{res(k)}(f)) \right) \\ &= \frac{3A}{m - Bk} \left(1 + \frac{Ak}{m - Bk} \right) F_1^{res(k)}(f)) \end{aligned}$$

This can be analyzed as follows:

$$\begin{aligned} (m - Bk)^2 - (Ak)^2 &\leq (m - Bk)^2 \\ (m - Bk + Ak)(m - Bk - Ak) &\leq (m - Bk)^2 \\ 1 + \frac{Ak}{m - Bk} &\leq \frac{(m - Bk)}{m - (A + B)k} \\ \frac{3A}{m - Bk} \left(1 + \frac{Ak}{m - Bk} \right) &\leq \frac{3A}{m - (A + B)k} \end{aligned}$$

Hence, we have a $(3A, A + B)$ guarantee for the k-tail estimation. □

In particular, since the two counter algorithms analyzed have k tail guarantees with constants $(1, 1)$, their summaries can be merged in this way to obtain k tail summaries with constants $(3, 2)$. Equivalently, this means to obtain a desired error Δ, we need to pick the number of counters m to be at most a constant factor (three) times larger to give the same bound on merging multiple summaries as for a single summary.

7. REFERENCES

[1] A. Arasu, S. Babu, and J. Widom. Cql: A language for continuous queries over streams and relations. *Proceedings of the 9th DBPL International Confenrence on Data Base and Programming Languages*, pages 1–11, 2003.

[2] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss. Combining geometry and combinatorics: a unified approach to sparse signal recovery. *Allerton*, 2008.

[3] R. Berinde, P. Indyk, and M. Ruzic. Practical near-optimal sparse recovery in the l_1 norm. *Allerton*, 2008.

[4] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. *Proceedings of 1999 ACM SIGMOD*, pages 359–370, 1999.

[5] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. *Proceedings of the 2nd IEEE MDM International Conference on Mobile Data Management*, pages 3–14, 2001.

[6] P. Bose, E. Kranakis, P. Morin, and Y. Tang. Bounds for frequency estimation of packet streams. *Proceedings of the 10th International Colloquium on Structural Information and Communication Complexity*, pages 33–42, 2003.

[7] E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1208–1223, 2006.

[8] A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for computing the entropy of a stream. In *SODA*, 2007.

[9] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Proceedings of the 29th ICALP International Colloqium on Automata, Languages and Programming*, pages 693–703, 2002.

[10] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. *PVLDB*, 1(2):1530–1541, 2008.

[11] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in data streams. *Proceedings of the 29th ACM VLDB International Conference on Very Large Data Bases*, pages 464–475, 2003.

[12] G. Cormode and S. Muthukrishnan. Improved data stream summaries: The count-min sketch and its applications. *FSTTCS*, 2004.

[13] E. Demaine, A. L. Ortiz, and J. Munro. Frequency estimation of internet packet streams with limited space. *Proceedings of the 10th ESA Annual European Symposium on Algorithms*, pages 348–360, 2002.

[14] D. L. Donoho. Compressed sensing. Unpublished manuscript, Oct. 2004.

[15] C. Estan and G. Verghese. New directions in traffic measurement and accounting. *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[16] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. Ullman. Computing iceberg queries efficiently. *Proceedings of the 24th ACM VLDB International Conference on Very Large Data Bases*, pages 299–310, 1998.

[17] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. One sketch for all: fast algorithms for compressed sensing. In *ACM STOC 2007*, pages 237–246, 2007.

[18] J. Han, J. Pei, G. Dong, and K. Wang. Efficient computation of iceberg cubes with complex measures. *Proceedings of 2001 ACM SIGMOD*, pages 1–12, 2001.

[19] J. Hershberger, N. Shrivastava, S. Suri, and C. D. Tóth. Space complexity of hierarchical heavy hitters in multi-dimensional streams. *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 338–347, 2005.

[20] P. Indyk. Algorithms for dynamic geometric problems over data streams. In *STOC*, 2004.

[21] P. Indyk. Sketching, streaming and sublinear-space algorithms. *Graduate course notes, available at* `http://stellar.mit.edu/S/course/6/fa07/6.895/`, 2007.

[22] P. Indyk and M. Ruzic. Near-optimal sparse recovery in the l_1 norm. *FOCS*, 2008.

[23] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)*, 28(1):51–55, 2003.

[24] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *VLDB*, pages 346–357, 2002.

[25] A. Metwally, D. Agrawal, and A. Abbabi. Efficient computation of frequent and top-k elements in data streams. *International Conference on Database Theory*, pages 398–412, 2005.

[26] J. Misra and D. Gries. Finding repeated elements. *Science of Computer Programming*, 2:142–152, 1982.

[27] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Foundations and Trends in Theoretical Computer Science, 2005.

[28] Compressed sensing resources. Available at `http://www.dsp.ece.rice.edu/cs/`, 2006. Rice DSP Group.

[29] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. *Proceedings of the 2nd International Conference on Embedded Network Sensor Systems*, pages 239–249, 2004.

[30] G. Zipf. *Human Behavior and The Principle of Least Effort*. Addison-Wesley, 1949.

APPENDIX

A. LOWER BOUND

THEOREM 13. *For any deterministic counter algorithm with m counters, for any k, $1 \leq k \leq m$, there exists some stream in which the estimation error of an element is at least $\frac{F_1^{res(k)}}{2m}$*

PROOF. The proof is similar to that of Theorem 2 in [6]. For some integer X, consider two streams A and B. The streams share the same prefix of size $X(m+k)$, where elements $a_1 \ldots a_{m+k}$ occur X times each. After the counter algorithm runs on this first part of each stream, only m elements can have non-zero counters. Assume without loss of generality that the other k elements are $a_1 \ldots a_k$.

Then stream A continues with elements $a_1 \ldots a_k$, while stream B continues with k other elements $z_1 \ldots z_k$ distinct from $a_1 \ldots a_{m+k}$. Both streams thus have total size $X(m+k)+k$.

For both streams, after processing the prefix of size $X(m+k)$, the algorithm has no record of any of the elements in the remaining parts of either of the streams. So the two remaining parts look identical to the algorithm and will yield the same estimates. Thus, for $1 \leq i \leq k$, $c_{a_i}(A) = c_{z_i}(B)$. But $f_{a_i}(A) = X+1$ while $f_{z_i}(B) = 1$. The counter error for one of the two streams must be at least $X/2$. Note that $F_1^{res(k)}(A) = Xm$ and $F_1^{res(k)}(B) = Xm + k$; then the error is at least

$$\frac{X}{2} \geq \frac{F_1^{res(k)}}{2m + 2k/X}$$

As $X \to \infty$, this approaches our desired bound. □

Thus an algorithm that provides an error bound of $\frac{F_1^{res(k)}}{m-k}$ must use at least $(m-k)/2$ counters.

B. TAIL GUARANTEE WITH CONSTANTS $A = B = 1$ FOR FREQUENT

We can interpret the FREQUENT algorithm in the following way: each element in the stream results in incrementing one counter; in addition, some number of elements (call this number d) also result in decrementing $m + 1$ counters (we can think of the d elements incrementing and later decrementing their own counter). The sum of the counters at the end of the algorithm is $\|c\|_1$. We have

$$\|c\|_1 = \|f\|_1 - d(m+1)$$

Since there were d decrement operations, and each operation decreases any given counter by at most one, it holds that the final counter value for any element is at least $f_i - d$. We restrict our attention to the k most frequent elements. Then

$$\begin{aligned} \|c\|_1 = \|f\|_1 - d(m+1) &\geq \sum_{i=1}^{k}(f_i - d) \\ \|f\|_1 - d(m+1) &\geq -dk + \sum_{i=1}^{k} f_i \\ \sum_{i=k+1}^{n} f_i &\geq d(m+1-k) \\ d &\leq \frac{F_1^{res(k)}}{m+1-k} \end{aligned}$$

Since the error in any counter is at most d, this implies the k-tail guarantee with $A = B = 1$.

C. TAIL GUARANTEE WITH CONSTANTS $A = B = 1$ FOR SPACESAVING

The tail guarantee follows almost immediately from the following claims proven in [25]:

LEMMA 3 IN [25]: *If the minimum non-zero counter value is Δ, then $\delta_i \leq \Delta$ for all i.*

THEOREM 2 IN [25]: *Whether or not element i (i.e. i-th most frequent element) corresponds to the i-th largest counter, the value of this counter is at least f_i, the frequency of i.*

If we restrict our attention to the k largest counters, the sum of their values is at least $\sum_{i=1}^{k} f_i$. Since in this algorithm the sum of the counters is always equal to the length of the stream, it follows that:

$$\Delta \leq \frac{\|f\|_1 - \sum_{i=1}^{k} f_i}{m-k}$$

thus by Lemma 3

$$\delta_i \leq \frac{F_1^{res(k)}}{m-k} \qquad \forall i$$

which is the k-tail guarantee with constants $A = B = 1$.